Heterogeneous and asynchronous networks of timed systems

Fiadeiro, José L.; Lopes, Antónia

Published in:
Theoretical Computer Science

DOI:
10.1016/j.tcs.2016.12.014

Publication date:
2017

Heterogeneous and Asynchronous Networks of Timed Systems

José L. Fiadeiro[a], Antónia Lopes[b]

[a] *Department of Computer Science, Royal Holloway University of London, UK*
[b] *Department of Informatics, Faculty of Sciences, University of Lisbon, Portugal*

Abstract

We present a component algebra and an associated logic for heterogenous timed systems that can be interconnected at run time. The components of the algebra are asynchronous networks of processes, where processes are sets of traces that model the behaviour of the software applications or devices that are interconnected and execute according to the clock granularity of the network node in which they are placed. The advantage of a trace-based model is that it abstracts from the specificities of the different classes of automata that can be chosen as models of implementations and characterises at a higher level the topological properties of the languages generated by such automata that support several compositionality results; in the paper, such properties are supported by a new time refinement relation and its related closure operator. The main novelty and contribution of our theory lies in the fact that we do not assume that all network nodes have the same clock granularity and that interconnections can be established, at run time, among nodes with different clock granularities. We investigate conditions under which the interconnected processes can communicate and make progress, generating a collective non-empty behaviour, i.e., conditions that ensure that the interconnection is consistent. Those conditions can be verified at design time, thus allowing that systems can be interconnected at run time without further checking for compatibility; to the best of our knowledge, no other component algebra has been put forward for timed heterogeneous systems that does not require *a-priory* knowledge of their structure. Finally, we propose a logic that can support specifications for this component algebra and prove associated compositionality results.

Keywords: asynchronous process networks, component algebra, heterogeneous time, orchestration, temporal logic

1. Introduction

The systems that are now operating in cyberspace are best modelled as networks of 'machines' (software applications, or devices such as sensor and actuators), where each machine performs local computations and can be interconnected dynamically (at run time) to other machines to achieve some goal. Systems such as these often have real-time requirements, i.e., their correctness depends not only on what outputs are returned to given inputs, but also on the time at which inputs are received and corresponding outputs are produced and communicated.

Because of the distributed nature of such networks, it does not make sense to assume that the nodes of the networks at which machines execute have the same clock granularity. This means that interconnections can only be established asynchronously (synchronicity being only possible if machines execute over compatible clocks) through connectors that can orchestrate the interactions between machines according to the clock granularities of the nodes at which they execute.

Several researchers have recently addressed discrete timed systems with heterogeneous clock granularities – an overview of related work is presented in Sec. 7. However, those approaches generally suffer from two major shortcomings:

(a) They do not provide a theory of *c*omposability for such systems that can support compositional reasoning about the properties of the composition operation, including that it preserves consistency, i.e., that the behaviour generated by a network is not empty, meaning that the machines can cooperate as interconnected.

(b) They rely on design-time modifications of the time structures to ensure compatibility, i.e., component implementations might have to be changed for the composition to work; this makes them unsuitable for addressing global properties of systems interconnected at run time as actually implemented.

In this paper, we put forward a component algebra for heterogeneous and asynchronous networks of timed systems that addresses both challenges: compositional reasoning and run-time interconnection. To the best of our knowledge, no other component algebra has been put forward for timed heterogeneous systems that does not require *a-priory* knowledge of their structure, i.e., that ensures run-time compositionality.

Our algebra abstracts the behaviour of machines as processes whose traces are generated according to the clock granularity of the network node in which they execute. The advantage of building the component algebra over sets of traces is that it makes our results independent of the specific choice of more operational abstractions that generate those behaviours, which in the literature on timed systems include, for example, discrete timed input/output automata (TIOA) [22, 9]. In order to illustrate how a specific type of operational model can be used in this context we use the version of TIOA defined in [11].

Our model is based on the component and interface algebra for service-oriented computing that we proposed in [13] for un-timed domains. A first extension of this model for timed systems was presented in [10] based on a (homogeneous) notion of time in which all processes execute according to the same time granularity. The present extension to a heterogeneous setting is not trivial (which justifies this paper) because, where the algebraic properties of composition in an homogenous time domain generalise those of the un-timed domain presented in [13], interconnection in a heterogeneous setting is much more involved — indeed, not even always admissible. More specifically, the main challenges come from:

(a) the fact that, in a heterogeneous timed domain, different clock granularities interfere with the way processes need to be coordinated in order to ensure that they can cooperate; and,

(b) the fact that the topological properties of timed traces are more intricate than those of un-timed ones, which means that we need to go beyond the usual Cantor topology of trace-based domains.

In relation to (a), we define in Sec. 3 a notion of connector through which processes can be interconnected into networks and a composition operator through which networks can be interconnected at run time. We investigate conditions that guarantee that every joint finite trace can be extended by an execution step (i.e., that the networks are 'progress-enabled'), i.e., that the machines can agree on what to execute next. Being progress-enabled is particularly important for run-time interconnection because it guarantees that, when binding to another process (or network), the current trace of a network can be extended to a trace of the newly formed network.

In relation to (b), we define in Sec. 2 a new time-related refinement relation and a new time-related closure operator that provide us with the topological properties required for characterising consistency of networks (Theo. 5.5), and for proving one of our major compositionality results (Theo. 5.8), which provides us sufficient conditions for ensuring that a network of progress-enabled processes is itself progress-enabled. Time refinement and its related closure operator are then investigated in Sec. 6 from the point of view of a suitable logic for our component algebra, including a compositionality property (Theo. 6.21).

The structure of the paper is as follows:

Section 2 introduces basic notions of trace-based semantic models, including the usual Cantor topology, and the new time-related refinement relation and closure operator. Notions of projection and translation are also introduced to account for interconnections.

Section 3 introduces the notion of process that we use to abstract from the behaviour on timed system components and, crucially, an explicit notion of connection through which process interconnections can be orchestrated.

Section 4 introduces the component algebra proper: the components are heterogeneous timed asynchronous relational nets – HT-ARNs – which are hypergraphs whose nodes are labelled with processes and whose hyperedges are labelled with connections; the composition operator allows two networks to be interconnected via interaction points. The fact that we work with a heterogeneous timed model implies a major departure in relation to traditional models of reactive systems such as process algebras, where composition results in a process; we define instead what it means for an HT-ARN to be approximated by a process and prove that for every connected HT-ARN there is a process that best approximates it (though the behaviour of the process does not necessarily coincide with that of the network). Such an approximation is useful to prove properties of the network and for simulating the network.

Section 5 investigates a major property of networks: consistency, i.e., the property that an HT-ARN has a non-empty set of behaviours meaning that its processes can co-operate as interconnected. We prove compositionality results for consistency through criteria that can be checked on processes at design time guaranteeing the consistency of interconnections when performed at run time across different clock granularities. We also show how such criteria can be checked over more operational abstractions of process implementations such as automata, and analyse the associated complexity.

Section 6 introduces a logic through which the behaviour of machines can be

specified or analysed. We abandon the implicit-time model used in [13], which is not realistic for the class of applications that need to run across heterogeneous time domains, in favour of a metric temporal logic [24]. The challenges here concern the need for topological notions of closure that go beyond the traditional safety-related ones [2], which leads us to investigate a continuous semantics with a new operator that captures the new notion of closure introduced in Sec. 2.

Finally, we provide an overview of related work (Sec. 7). Proofs of our main results are collected in an appendix at the end of the paper.

This paper is itself a revised and extended version of [14]. More specifically,

(a) we expand on the topological properties and constructs that are used for defining the component algebra and characterising consistency;

(b) we use deterministic timed input-output machines to illustrate how a specific class of machines can be used as implementations of our notions of process and of connector, and prove compositionality results for that class; and

(c) we revise the semantics of the continuous temporal logic used therein for specification, investigate a pointwise (discrete) semantics, and develop a new fragment in which the continuous and pointwise semantics coincide; in this fragment, we characterise satisfiability of specifications and the existence of canonical models.

2. Preliminaries

The processes that execute in cyberspace are typically open, reactive and interactive. Their behaviour can be observed in terms of the actions that they perform. For simplicity, we use a linear time model, i.e., we observe streams of actions. In order not to constrain the environment in which processes execute and communicate, we take streams that capture complete behaviours to be infinite (which we call traces) and we allow several actions to occur 'simultaneously', i.e., the granularity of observations may not be so fine that we can always tell which of two actions occurred first. The execution of an empty set of actions corresponds to a step during which a process is idle, i.e., a step performed by the environment without the involvement of the process.

The following definition sets out terminology and notation that is used throughout the paper. We start by recalling a few standard concepts related to traces and their Cantor topology.

Definition 2.1 (Trace, segment, property, closure)**.** *Let* S *be a set.*

- *A* trace λ *over* S *is an element of* S^ω*, i.e., an infinite sequence of elements of* S*. We denote by* λ_i *the prefix of* λ *that ends at* $\lambda(i-1)$ *if* $i > 0$*, with* λ_0 *being the empty sequence.*

- *A* segment π *is an element of* S^**, i.e., a finite sequence of elements of* S*, the length of which we denote by* $|\pi|$*. We use* $\pi < \lambda$ *to mean that the segment* π *is a prefix of* λ*. Given* $s \in \mathsf{S}$*, we denote by* $\pi \cdot s$ *the segment obtained by extending* π *with* s*.*

- *A* property Λ *over* S *is a set of traces. For every property* Λ*, we define:*

- $\downarrow\Lambda = \{\pi : \exists\lambda \in \Lambda(\pi < \lambda)\}$ — *the segments that are prefixes of traces in* Λ*, also called the* downward closure *of* Λ*.*
- $\bar{\Lambda} = \{\lambda : \forall\pi < \lambda(\pi \in \downarrow\Lambda)\}$ — *the traces whose prefixes are in* $\downarrow\Lambda$*, also called the* closure *of* Λ*.*

- *A property* Λ *is said to be* closed *iff* $\Lambda \supseteq \bar{\Lambda}$ *(and, hence,* $\Lambda = \bar{\Lambda}$*).*

In our model, timed traces consist of an infinite sequence of pairs of a set of actions and an instant of time — the actions that are observed at that instant, typically performed by different components of a network. That is, henceforth we work with traces over $\mathsf{S} = 2^{\mathsf{A}} \times \mathbb{R}_{\geq 0}$ where A is a set (of actions) and $\mathbb{R}_{\geq 0}$ is the set of the non-negative real numbers. Notice that any such trace is uniquely defined by a pair consisting of a trace over 2^{A} and a trace over $\mathbb{R}_{\geq 0}$.

Definition 2.2 (Timed traces)**.** *Let* A *be a finite set (of actions).*

- *A* time sequence τ *is a trace over* $\mathbb{R}_{\geq 0}$ *(the non-negative real numbers) such that:* $\tau(0) = 0$*;* $\tau(i) < \tau(i+1)$ *for every* $i \in \mathbb{N}$*; the set* $\{\tau(i) : i \in \mathbb{N}\}$ *is unbounded, i.e., time progresses (the 'non-Zeno' condition).*

- *An* action sequence σ *is a trace over* 2^{A} — *i.e., an infinite sequence of sets of actions — such that* $\sigma(0) = \emptyset$*.*

- *A* timed trace *over* A *is a trace* $\langle\sigma, \tau\rangle$ *over* $2^{\mathsf{A}} \times \mathbb{R}_{\geq 0}$ *such that* σ *is an action sequence and* τ *is a time sequence. We denote by* $\mathsf{ttra}(\mathsf{A})$ *the set of timed traces over* A*.*

- *A* timed segment *over* A *is a segment* $\langle\pi, \tau\rangle$ *over* $2^{\mathsf{A}} \times \mathbb{R}_{\geq 0}$ *such that, if the segment is not empty,* $\pi(0) = \emptyset$*,* $\tau(0) = 0$*, and* $\tau(i) < \tau(i+1)$ *for every* $i < |\tau| - 1$*. We denote by* $\mathsf{tseg}(\mathsf{A})$ *the set of timed segments over* A*.*

- *A* timed property *over* A *is a subset of* $\mathsf{ttra}(\mathsf{A})$*. Given a timed property* Λ*, we define:*

  - $\downarrow\Lambda = \{\pi \in \mathsf{tseg}(\mathsf{A}) : \exists\lambda \in \Lambda(\pi < \lambda)\}$.
  - $\bar{\Lambda} = \{\lambda \in \mathsf{ttra}(\mathsf{A}) : \forall\pi < \lambda(\pi \in \downarrow\Lambda)\}$.

  That is, we restrict closure as introduced in Def. 2.1 to the sub-space of timed traces and timed segments.

- *Given* $\delta \in \mathbb{R}_{>0}$*,*

  - *The* δ-time sequence τ_δ *is defined by* $\tau_\delta(i) = i \cdot \delta$ *for every* $i \in \mathbb{N}$*, i.e., it consists of all multiples of* δ*.*
  - *A* δ-timed trace *over* A *is a timed trace* $\langle\sigma, \tau_\delta\rangle$*, the set of which is denoted by* $\mathsf{ttra}_\delta(\mathsf{A})$*.*
  - *A* δ-timed property *is a timed property that consists of* δ*-timed traces.*

Notice that, by allowing sets of actions to be empty, we can model finite behaviours through timed traces that, after some point, have only the empty set, i.e., which are of the form $\langle\pi.\emptyset^\omega, \tau\rangle$ where $\pi \in (2^{\mathsf{A}})^*$ — the system stops executing actions after a certain point in time whilst still part of a network.

The empty set also allows us to model observations that are triggered by actions performed by components outside the system.

This time model falls under what is often known as a 'point-based semantics', as opposed to an 'interval-based semantics' in which observations are made at every instant of time — our systems are discrete and, therefore, a continuous observation model is not required. The advantages of the adopted model are that, on the one hand, it offers a natural extension of a trace-based model (as adopted, for example, in [13]) and, on the other hand, it has been recently studied from the point of view of a number of decidability results [26].

This model is still realistic for hybrid systems of software and physical devices: typically, a physical device interacts with software applications via a 'sampler', a component that observes the state of the physical device at some periodical rate δ and communicates the observations to software components of the system (such as a controller) [17]. In other words, for modelling cyberphysical systems, we take into account the discrete-time behaviour that results from the interaction of physical devices with their samplers.

The closure operator introduced in Def. 2.1 can be applied directly to $\mathsf{S} = 2^{\mathsf{A}} \times \mathbb{R}_{\geq 0}$, i.e., to the Cantor topology defined over timed traces. However, it is often useful to separate properties required of action sequences from those of time sequences, which is why we define a different operator for timed properties that uses the closure of operator over $(2^{\mathsf{A}})^{\omega}$ for fixed time sequences.

Definition 2.3 (t-closure)**.** *Given a timed property Λ over A we define,*

- *for every time sequence τ, $\Lambda_\tau = \{\sigma \in (2^{\mathsf{A}})^{\omega} : \langle \sigma, \tau \rangle \in \Lambda\}$ — the* action property *defined by Λ and τ*
- $\Lambda_{time} = \{\tau : \exists \sigma \in (2^{\mathsf{A}})^{\omega} (\langle \sigma, \tau \rangle \in \Lambda)\}$ — *the time sequences of traces in Λ*
- $\Lambda^t = \bigcup_{\tau \in \Lambda_{time}} \{\langle \sigma, \tau \rangle : \sigma \in \overline{(\Lambda_\tau)}\}$ — *the* t-closure *of Λ*

We say that Λ is closed relative to time *or, simply, t-closed, iff $\Lambda \supseteq \Lambda^t$ (and, hence, $\Lambda = \Lambda^t$).*

In order to be able to model networks of systems whose nodes have different time granularities, we need a notion of time refinement through which behaviours observed at different nodes can be brought down to a finer time granularity:

Definition 2.4 (Time refinement)**.** *Let $\rho : \mathbb{N} \to \mathbb{N}$ be a monotonically increasing function that satisfies $\rho(0) = 0$.*

- *Let τ, τ' be two time sequences. We say that τ'* refines *τ through ρ, which we denote by $\tau' \preceq_\rho \tau$, iff, for every $i \in \mathbb{N}$, $\tau(i) = \tau'(\rho(i))$. We say that τ'* refines *τ, which we denote by $\tau' \preceq \tau$, iff $\tau' \preceq_\rho \tau$ for some ρ.*
- *Let $\lambda = \langle \sigma, \tau \rangle$, $\lambda' = \langle \sigma', \tau' \rangle$ be two timed traces. We say that λ' refines λ through ρ, which we denote by $\lambda' \preceq_\rho \lambda$, iff*
  - *$\tau' \preceq_\rho \tau$*
  - *for every $i \in \mathbb{N}$ and $\rho(i) < j < \rho(i{+}1)$, $\sigma(i) = \sigma'(\rho(i))$ and $\sigma'(j) = \emptyset$.*

  We also say that λ' refines λ, which we denote by $\lambda' \preceq \lambda$, iff $\lambda' \preceq_\rho \lambda$ for some ρ.

- *The r-closure of a timed property* Λ *over* A *is* $\Lambda^r = \{\lambda' \in \mathsf{ttra}(\mathsf{A}) : \exists\lambda{\in}\Lambda(\lambda' \preceq \lambda)\}$. *We say that* Λ *is* closed under time refinement, *or r-closed, iff* $\Lambda^r \subseteq \Lambda$.

We extend the notion of refinement to timed properties:

- *A timed property* Λ' *refines a timed property* Λ — $\Lambda' \preceq \Lambda$ — *if, for every* $\lambda'{\in}\Lambda'$, *there exists* $\lambda{\in}\Lambda$ *such that* $\lambda' \preceq \lambda$.
- *A timed property* Λ' approximates *a timed property* Λ — $\Lambda' \precapprox \Lambda$ — *if* $\Lambda' \preceq \Lambda$ *and, for every* $\lambda{\in}\Lambda$, *there exists* $\lambda'{\in}\Lambda'$ *such that* $\lambda' \preceq \lambda$.

A time sequence refines another if the former interleaves time observations between any two time observations of the latter. Refinement extends to traces by requiring that no actions be observed in the finer trace between two consecutive times of the coarser (see Fig. 1 and also Fig. 5).

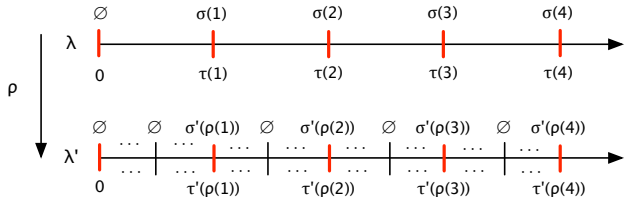

Figure 1: Time refinement – for every $i \in \mathbb{N}$, $\tau(i) = \tau'(\rho(i))$ and $\sigma(i) = \sigma'(\rho(i))$

Proposition 2.5. *The following properties of time refinement are useful:*

- *Given* $\tau' \preceq_\rho \tau$ *and a timed trace* $\lambda = \langle\sigma, \tau\rangle$, *there is a single timed trace* $\lambda' = \langle\sigma', \tau'\rangle$ *such that* $\lambda' \preceq_\rho \lambda$ — *we call* λ' *the refinement of* λ *over* τ'.
- $\tau_{\delta'} \preceq \tau_\delta$ *iff* δ *is a multiple of* δ'.
- *If two timed properties* Λ_1 *and* Λ_2 *are r-closed so is their intersection* $\Lambda_1 \cap \Lambda_2$.
- *If* Λ *is r-closed,* $\Lambda' \preceq \Lambda$ *iff* $\Lambda' \subseteq \Lambda$.

It is not difficult to prove that the refinement relation makes the space of all time sequences a complete meet semi-lattice. The meet of two time sequences τ_1 and τ_2 always exists and is given by the recursion

$$\tau(i+1) = min(\{\tau_1(j) > \tau(i), j \in \mathbb{N}\} \cup \{\tau_2(j) > \tau(i), j \in \mathbb{N}\})$$

together with the base $\tau(0) = 0$. However, if one considers the space of all time sequences of the form τ_δ for some $\delta{\in}\mathbb{R}_{>0}$, it is easy to see that a meet of τ_{δ_1} and τ_{δ_2} exists iff δ_1 and δ_2 are commensurate (have a common divisor), i.e., if there are $n, m{\in}\mathbb{N}_{>0}$ such that $\delta_1/n = \delta_2/m$, in which case the meet is τ_δ where δ is their greatest common divisor (which always exists and can be calculated using Euclid's algorithm provided that δ_1 and δ_2 are commensurate).

Functions between sets of actions *(alphabet maps)* are useful for defining relationships between individual machines and the networks in which they operate:

Definition 2.6 (Projection and translation)**.** *Let* $f : \mathsf{A} \to \mathsf{B}$ *be a function (alphabet map).*

- *For every* $\sigma \in (2^{\mathsf{B}})^\omega$*, we define* $\sigma|_f \in (2^{\mathsf{A}})^\omega$ *pointwise as* $\sigma|_f(i) = f^{-1}(\sigma(i)) = \{a \in \mathsf{A} : f(a) \in \sigma(i)\}$ *— the* projection *of* σ *over* A*. If* f *is an inclusion* ($\mathsf{A} \subseteq \mathsf{B}$)*, then we tend to write* $\_|_{\mathsf{A}}$ *instead of* $\_|_f$*.*
- *For every timed trace* $\lambda = \langle \sigma, \tau \rangle$ *over* B*, we define its projection over* A *to be* $\lambda|_f = \langle \sigma|_f, \tau \rangle$ *and, for every timed property* Λ *over* B*, we define* $\Lambda|_f = \{\lambda|_f : \lambda \in \Lambda\}$ *— the* projection *of* Λ *to* A*.*
- *For every timed property* Λ *over* A*, we define* $f(\Lambda) = \{\langle \sigma, \tau \rangle : \langle \sigma|_f, \tau \rangle \in \Lambda\}$ *— the* translation *of* Λ *to* B*.*
- *For every timed property* Λ_{A} *over* A *and every timed property* Λ_{B} *over* B *we write* $\Lambda_{\mathsf{B}} \preceq^f \Lambda_{\mathsf{A}}$ *(resp.* $\Lambda_{\mathsf{B}} \precapprox^f \Lambda_{\mathsf{A}}$*) to mean* $\Lambda_{\mathsf{B}} \preceq f(\Lambda_{\mathsf{A}})$ *(resp.* $\Lambda_{\mathsf{B}} \precapprox f(\Lambda_{\mathsf{A}})$*).*

That is, $\sigma|_f$ projects every trace σ over B to a trace over A by taking the inverse image of the set of actions at every point of σ. An inclusion $\mathsf{A} \subseteq \mathsf{B}$ defines a function that maps every element in itself. We denote by $\_|_{\mathsf{A}}$ the corresponding projection of traces over B to traces over A, which forgets the actions of B that are not in A. An example is given in Sec. 3.2. Notice that the inverse image of a set of actions in B that are not in the range of f is the empty set; if f captures the way in which a machine is part of a network, this means that sets of actions of the network in which the machine is not involved are projected to the machine as $\emptyset$.

A timed property Λ over A is mapped forwards to a timed property over B by taking the set of all traces over B that are projected back to a trace of Λ. Notice that this is different from applying f pointwise to every trace λ of Λ: instead, our construction maps $\lambda = \langle \sigma, \tau \rangle$ to all traces $\langle \sigma', \tau \rangle$ over B such that, for all i, $\sigma(i) = f^{-1}(\sigma'(i))$, which means that every $\sigma'(i)$ may contain any actions of B that are not in the range of f. In particular, we have that $f(\mathsf{ttra}(\mathsf{A})) = \mathsf{ttra}(\mathsf{B})$. Again, if f captures the way in which a machine is part of a network, $f(\Lambda)$ will open every trace of Λ to actions of the network in which the machine is not involved. This is essential for defining the semantics of networks (which we do in Sec. 4.1).

The following proposition provides useful properties: that projections preserve time refinement and translations preserve r-closure and t-closure.

Proposition 2.7 (Preservation)**.** *Let* $f : \mathsf{A} \to \mathsf{B}$ *be a function (alphabet map).*

- *Given two timed traces* λ *and* λ' *over* B *such that* $\lambda' \preceq_\rho \lambda$*,* $\lambda'|_f \preceq_\rho \lambda|_f$*.*
- *Let* Λ *be a timed property over* B*. If* Λ *is r-closed, so is* $\Lambda|_f$*.*
- *Let* Λ *be a timed property over* A*. If* Λ *is r-closed, so is* $f(\Lambda)$*.*
- *Let* Λ *be a timed property over* A*. If* Λ *is t-closed , so is* $f(\Lambda)$*.*

We are particularly interested in translations defined by prefixing every element of a set with a given symbol. Such translations are useful for identifying in a network the machine to which an action belongs — because they can bind to other machines at run time, not design time, it would not be realistic to assume that machines have mutually disjoint alphabets. More precisely, given a set A

and a symbol p, we denote by $(p.\_)$ the function that prefixes the elements of A with '$p.$'. Note that prefixing defines a bijection between A and its image $p.\mathsf{A}$.

3. Processes and Connections

In this section, we put forward a component algebra for heterogenous timed systems. Components are networks of processes (not individual processes) and the composition operator of the algebra creates complex networks out of simpler ones. This is important to support modern systems of systems that operate in cyberspace, where systems (networks of simpler components) can bind at run time to other systems to obtain required resources or to jointly achieve some goal.

This framework generalises the component algebra proposed in [13] for service-oriented systems. The main differences are that (1) we address networks of processes that operate over heterogeneous time, and (2) we generalise the interconnection and coordination model to account for multi-party, not just peer-to-peer interactions.

We start by detailing the communication model and then proceed to defining networks and investigating some of their properties.

3.1. Processes

Processes are behavioural abstractions of units of computation and communication (machines) that execute at network nodes. Our communication model is asynchronous, interactions between machines being based on the exchange of messages; as explained in the previous section, in the case of cyberphysical systems, we assume that physical devices interface with software components via samplers that observe the states of the devices and publish their observations as messages.

We organise messages in sets that we call ports: a *port* is a finite set (of messages). Ports are communication abstractions that are convenient for organising networks of systems as formalised below. Every message belonging to a port has an associated *polarity*: $^-$ if it is an outgoing message (published at the port) and $^+$ if it is incoming (delivered at the port); this is the notation used in [6], among others. Therefore, every port M has a partition $M^- \cup M^+$. For every port M we define its dual M^{op}, which is obtained by swapping the polarities of the messages in M, i.e., $M^{op^-} = M^+$ and $M^{op^+} = M^-$.

The actions of sending (publishing) or receiving (being delivered) a message m are denoted by $m!$ and $m¡$, respectively. More specifically, if M is a port, we define:

- $\mathsf{A}_{M^-} = \{m! : m \in M^-\}$ — the set of publications (of outgoing messages) associated with M
- $\mathsf{A}_{M^+} = \{m¡ : m \in M^+\}$ — the set of deliveries (of incoming messages) associated with M
- $\mathsf{A}_M = \mathsf{A}_{M^-} \cup \mathsf{A}_{M^+}$ — the set of actions associated with M

Note that even if a process does not refuse the delivery of messages, it can decide to discard them, for example if they arrive outside the expected protocol. Not all published messages can be guaranteed to be delivered to their destination either,

i.e., we do not make any default assumptions on the transmission protocols through which processes exchange messages.

Definition 3.1 (Process)**.** *A* process *is a triple* $P = \langle \delta, \gamma, \Lambda \rangle$ *where:*

- $\delta \in \mathbb{R}_{>0}$ *is the granularity of the clock of the process;*
- γ *is a finite set of mutually disjoint ports;*
- Λ *is the r-closure of a non-empty* δ*-timed property over* $\mathsf{A}_\gamma = \bigcup_{M \in \gamma} \mathsf{A}_M$*.*

We call A_γ *the* alphabet *of the process and* Λ *the* behaviour *of the process.*

The fact that processes are r-closed means that they contain all possible interleavings of empty observations, thus capturing their behaviour in any possible environment. This notion of closure can be related to mechanisms that, such as stuttering [1], ensure that components do not constrain their environment.

Given a port M, we designate the process $\langle \delta, \{M\}, \mathsf{ttra}_\delta(\mathsf{A}_M)^r \rangle$ by $\boxdot^\delta_M$. This is a process with a single port M that, at any multiple of its clock granularity, accepts any set of actions belonging to A_M, which in the literature, and henceforth, is named RUN (see Fig. 2).

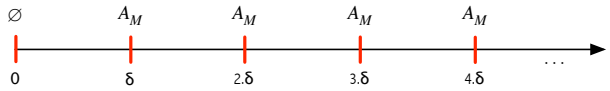

Figure 2: One of the top traces of $\boxdot^\delta_M$.

Another example of a process, which we will use later on, is *CreditRequest* $= \langle \delta, \gamma, \Lambda \rangle$ where:

- $\delta = 0.5$
- $\gamma = \{M\}$ where $M^- = \{creditReq, accept\}$, i.e. the process can send *creditReq* and *accept*, and $M^+ = \{approved, denied, transferDate\}$, i.e., it can receive *approved*, *denied* and *transferDate*.
- Λ consists of the closure of all those 0.5-timed traces where the process starts by sending *creditReq*, after which it waits up to ten time units for receiving *approved* or *denied*; in the first case, it sends *accept* and waits up to fifty time units for receiving *transferDate*, after which it stops; in the second case, it stops. An example of such a trace is depicted in Fig. 3.

3.2. Attachments and connections

Our model of interaction is based on orchestrating the joint behaviour of a collection of parties, each of which defines a process; the same party may engage in different orchestrations. Each such orchestration is performed by another process – the orchestrator – that synchronises separately with each party to coordinate their joint behaviour. Each party is connected to the orchestrator by what we call an attachment:

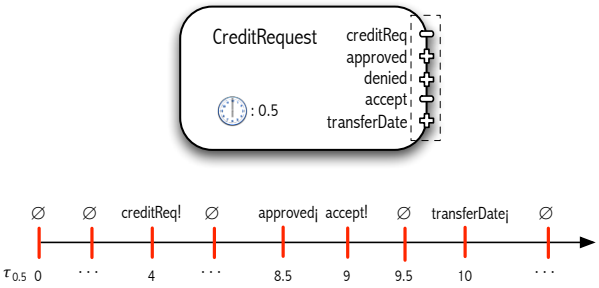

Figure 3: Process *CreditRequest* and one of its top traces.

Definition 3.2 (Attachment)**.** *An* attachment *is a triple* $\langle C, \xi : M_C \to M_P, P\rangle$ *where* $C = \langle \delta_C, \gamma_C, \Lambda_C\rangle$ *and* $P = \langle \delta_P, \gamma_P, \Lambda_P\rangle$ *are processes, and* ξ *is a bijection between two ports* $M_C \in \gamma_C$ *and* $M_P \in \gamma_P$ *that reverses polarities, i.e.,* $\xi(M_C^+) = M_P^-$ *and* $\xi(M_C^-) = M_P^+$*.*

An attachment is well formed *iff* δ_P *is a multiple of* δ_C*.*

We often use ξ *to designate the whole attachment (triple) if it is clear from the context which are the processes involved.*

Notice that ξ induces a translation ξ^{a} between A_{M_C} and A_{M_P} by switching publications and deliveries, i.e., $\xi^{\mathsf{a}}(m¡) = \xi(m)!$ for $m \in M_C^+$ and $\xi^{\mathsf{a}}(m!) = \xi(m)¡$ for $m \in M_C^-$ — this is because what one party sends, the other receives, and vice-versa. The condition that δ_P is a multiple of δ_C for the attachment to be 'well formed' reflects the fact that the source C (the orchestrator) needs to be able to 'tick' (deliver and receive messages) in a way that is compatible with the target P (the process that is a party in the orchestration).

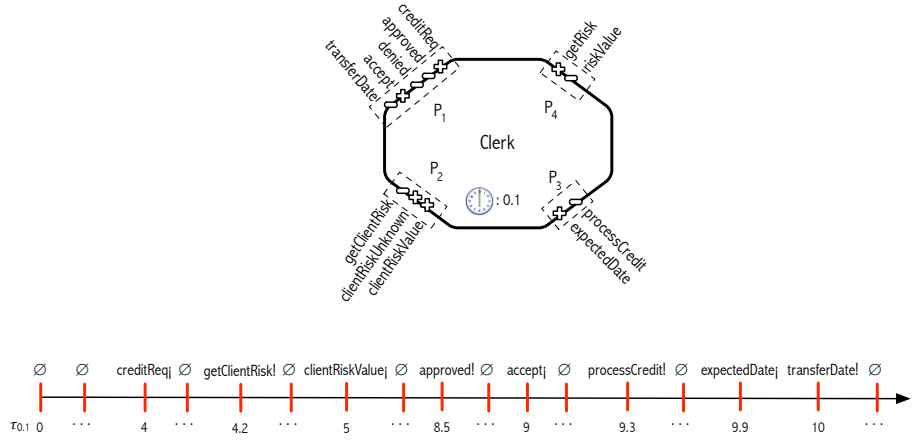

Figure 4: The orchestrator *Clerk* and one of its top traces.

As an example, consider the process (an orchestrator) *Clerk* depicted in Fig. 4. This process has four ports: P_1, P_2, P_3, P_4. For instance, in port P_1, it receives messages *creditReq* and *accept* and sends *approved*, *denied* and *transferDate*. Its clock rate is 0.1. After the delivery of the first *creditReq* on

P_1, it publishes *getClientRisk* on P_2 within five time units; then it waits for the delivery of *clientRiskValue* or *clientRiskUnknown* in the same port. If the risk of the transaction is known, this is enough for making a decision and sending *approved* or *denied* in port P_1 within 10 time units. After sending *approved*, *Clerk* waits at most five time units for the delivery of *accept* on P_1, upon which it publishes *processCredit* on P_3 within three time units and waits for the delivery of *expectedDate* on the same port; when this happens, it sends *transferDate* on P_1 within one time unit.

The identity between the unique port of *CreditRequest* and the port P_1 of *Clerk* establishes an attachment of *Clerk* to *CreditRequest*. The attachment is well formed because the clock rate of *CreditRequest*(0.5) is a multiple of that of *Clerk*(0.1).

Notice that, because *CreditRequest* has only one port (the one through which it is attached to *Clerk*), the attachment defines a (polarity reversing) mapping from the alphabet of *CreditRequest* to that of *Clerk*. It is easy to see that the projection (see Def. 2.6) of the trace of *Clerk* in Fig. 4 along this mapping is a refinement of that of *CreditRequest* in Fig. 3 and, hence, belongs to *CreditRequest* (processes being r-closed). This is depicted in Fig. 5.

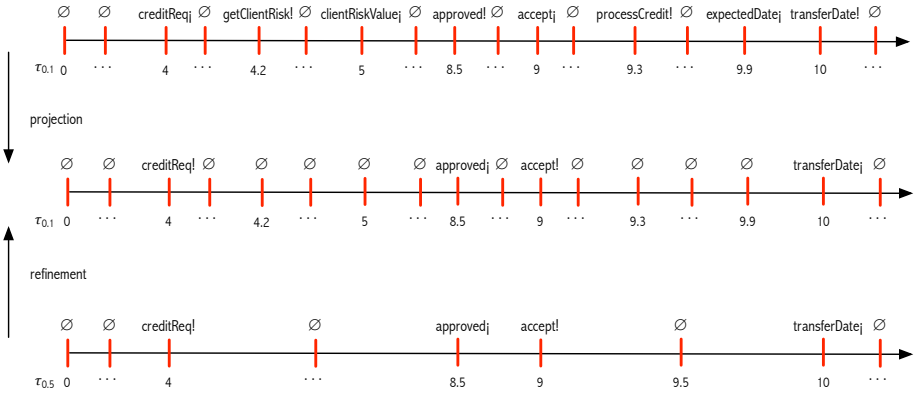

Figure 5: The middle trace is a polarity-reversing projection of the trace depicted in Fig. 4 and a refinement of the trace depicted in Fig. 3.

Attachments are used for building connections:

Definition 3.3 (Connection)**.** *A* connection *is a triple* $\Xi = \langle C, \gamma_F, \xi \rangle$ *where:*

- C *is a process* $\langle \delta, \gamma, \Lambda \rangle$ — *the orchestrator of the connection;*
- $\gamma_F \subseteq \gamma$ *consists of the ports of* C *that are 'free';*
- ξ *assigns to each* $M \in \gamma$ *a well-formed attachment* $\langle C, \xi_M, P_M \rangle$ *for some process* P_M *such that,*
  - *if* $M \neq M'$ *and* $P_M = P_{M'}$*, then* ξ_M *and* $\xi_{M'}$ *have different target ports, i.e., if two attachments connect* C *to the same process* P *then they use different ports of* P
  - *for every* $M \in \gamma_F$*,* $P_M = \boxdot^{\delta}_{M^{op}}$ *and* ξ_M *is the identity, i.e., the free ports are attached to* RUN*.*

That is, a connection consists of a process that orchestrates interactions among a number of parties. Those parties are attached to the orchestrator, not directly to each other, thus making communication between parties to be asynchronous. Some of the ports of the orchestrator may be 'free', thus accounting for the ability of the connection to grow at run time by accepting new parties, i.e., connections may be open. Those free ports are attached to RUN — a 'dummy' process that exhibits any possible behaviour — precisely so that it can be replaced at run time by other processes. Each port of a party can only be used by at most one attachment, i.e., if a party plays different roles in the same connection, it does so via different ports.

Because all the attachments in a connection need to be well formed, the clock granularity of each party P_M needs to be a multiple of that of the orchestrator C. Therefore, not all sets of processes can be interconnected: in order to be part of a connection, their clock rates need to have a common divisor, precisely so that they can interact via the orchestrator.

As an example, consider the connection depicted in Fig. 6, where *Clerk* orchestrates *CreditRequest* and two other processes, all the attachments being identities:

- *ClientsDB* is a process that gets information on risk from a database of clients. It has a single port through which it receives *getClientRisk* and sends *clientRiskValue* and *clientRiskUnknown*. When the first *getClientRisk* is delivered, it takes no more than seven time units to publish either *clientRiskValue* or *clientRiskUnknown*. The granularity of its clock is 0.2.

- *CreditMgr* handles approved credit requests. It has a single port through which it receives *processCredit* and sends *expectedDate*. When the first *processCredit* is delivered, *CreditMgr* takes no more than four time units to publish *expectedDate*. The granularity of its clock is 0.3.

The fact that in the connection the port P_4 is free (which is represented in Fig. 6 by a grey shadow) means that, should *Clerk* need to assess the risk of a non-client, another network would have to be found that can be attached, through that port, to *Clerk*. The idea is that the discovery and binding can be made at run time, possibly based on additional information that may be available about the non-client. This will be discussed in the next section.

4. Heterogeneous Timed Asynchronous Relational Nets

4.1. Networks

The model that we present for networks of timed systems generalises the notion of asynchronous relational network (ARN) proposed in [13] so as to capture a larger class of systems where coordination of interactions takes place among groups of processes, not just between pairs of processes (which are typical of service-oriented computing). This extension requires a different algebraic structure for the networks, which is why we adopt hypergraphs instead of simple graphs as the underlying mathematical structure.

Very briefly, a finite hypergraph is a pair $\langle N, E\rangle$ where N is a non-empty finite set of nodes and E is a finite set of hyperedges, each hyperedge being a non-empty set of nodes (in a graph, an edge is a pair of nodes). We label nodes

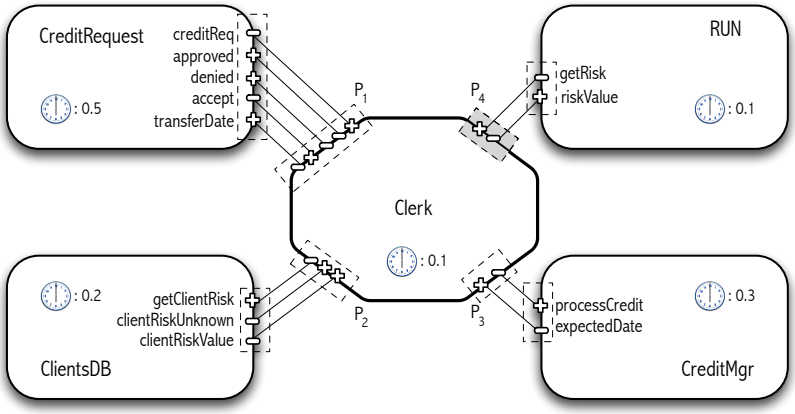

Figure 6: A connection with a free port and three processes.

with processes and hyperedges with connections that attach the processes at the nodes to the orchestrator of the connection.

Definition 4.1 (HT-ARN)**.** *A* heterogenous timed asynchronous relational net *(*HT-*ARN) α is a tuple $\langle N, E, \Omega, \Xi \rangle$ where:*

- $\langle N, E \rangle$ *is a finite hypergraph.*
- Ω *is a labelling function that, to every $p \in N \cup E$ (i.e., to every node and hyperedge) assigns a process $\Omega_p = \langle \delta_p, \gamma_p, \Lambda_p \rangle$*
- Ξ *is a labelling function that, to every $c \in E$ (i.e., to every hyperedge) assigns a connection $\Xi_c = \langle \Omega_c, \gamma_{cF}, \xi_c \rangle$ such that:*

  i) For every hyperedge c, we have an onto mapping κ_c from the set γ_c of ports of Ω_c to c, i.e., each of its nodes is associated with at least one of the ports of the orchestrator Ω_c.

  ii) For every hyperedge c and port $M \in \gamma_c$, ξ_c assigns an attachment $\langle \Omega_c, \xi_{cM} : M \to M', \Omega_{\kappa_c(M)} \rangle$ between the orchestrator Ω_c and the process $\Omega_{\kappa_c(M)}$.

  iii) If two hyperedges c_1 and c_2 share a node p then, for any ports $M_1 \in \gamma_{c_1}$ and $M_2 \in \gamma_{c_2}$ such that $\kappa_{c_1}(M_1) = \kappa_{c_2}(M_2) = p$, $\xi_{c_1 M_1}$ and $\xi_{c_2 M_2}$ have different codomains, i.e., attach to different ports of Ω_p.

We also define the following sets and mappings:

- $\mathsf{A}_\alpha = \bigcup_{p \in N} p.\mathsf{A}_{\gamma_p}$ *is the alphabet associated with α — the union of the alphabets of the processes that label the nodes translated by prefixing all actions with the corresponding node (which ensures that the union is disjoint).*

  iv) For every node $p \in N$, we denote by ι_p the function that maps A_{γ_p} to A_α, which prefixes the actions of A_{γ_p} with p.

  v) For every hyperedge $c \in E$, we denote by ι_c the function that maps A_{γ_c} to A_α. This function is such that, for every $M \in \gamma_c$, $\iota_c(\mathsf{A}_M) =$

$\iota_{\kappa_c(M)}(\xi^a_{c_M}(\mathsf{A}_M))$. *That is, actions of the orchestrator that belong to a port* M *are translated through* $\xi^a_{c_M}$ *to the attached process* $\kappa_c(M)$ *(reversing polarities) and then according to* $\iota_{\kappa_c(M)}$.

- $\Lambda_\alpha = \{\lambda \in \mathsf{ttra}(\mathsf{A}_\alpha) : \forall p \in N \cup E \ (\lambda|_{\iota_p} \in \Lambda_p)\}$

Note that, for every $p{\in}N$, $(\_|_{\iota_p})$ first removes the actions that are not in the alphabet $p.\mathsf{A}_p$ and then removes the prefix p, and similarly for every $p \in E$. Therefore, the set Λ_α consists of all traces over the alphabet of the HT-ARN that are projected to traces of all its processes and orchestrators:

$$\Lambda_\alpha = \textstyle\bigcap_{p\in N\cup E} \iota_p(\Lambda_p)$$

We take this set to represent the behaviour of α. That is, the behaviour of the HT-ARN is given by the intersection of the behaviour of the processes at the nodes and the hyperedges (connections) translated to the alphabet of the HT-ARN — this corresponds to what one normally understands as a parallel composition in trace-based models. This is justified by the fact that, when applied to a set of traces, the translations effectively open the behaviour of the processes to actions in which they are not involved (see the discussion after Def. 2.6).

As an example, consider the HT-ARN whose hypergraph has $\{c, d, m, r\}$ as its set of nodes and as its single hyperedge (that is, the hyperedge connects all nodes). The labelling function is, on nodes, c:*CreditRequest*, d:*ClientsDB*, m:*CreditMgr* and r:RUN, and assigns the connection $\langle \mathit{Clerk}, \{P_4\}, \{\xi_{P_1}, \ldots, \xi_{P_4}\}\rangle$ in Fig. 6 to the hyperedge, with $\xi_{P_1} = \langle \mathit{Clerk}, \mathit{id}, \mathit{ClientsDB}\rangle$, $\xi_{P_2} = \langle \mathit{Clerk}, \mathit{id}, \mathit{CreditRequest}\rangle$, $\xi_{P_3} = \langle \mathit{Clerk}, \mathit{id}, \mathit{CreditMgr}\rangle$ and $\xi_{P_4} = \langle \mathit{Clerk}, \mathit{id}, \mathrm{RUN}\rangle$. The HT-ARN itself is depicted in Fig. 7, the difference from the previous figure being that the nodes of the HT-ARN are represented explicitly, i.e., the figure depicts an hypergraph. Note that *Clerk* is not labelling a node (the hypergraph has four nodes, not five) but part of the label of the hyperedge.

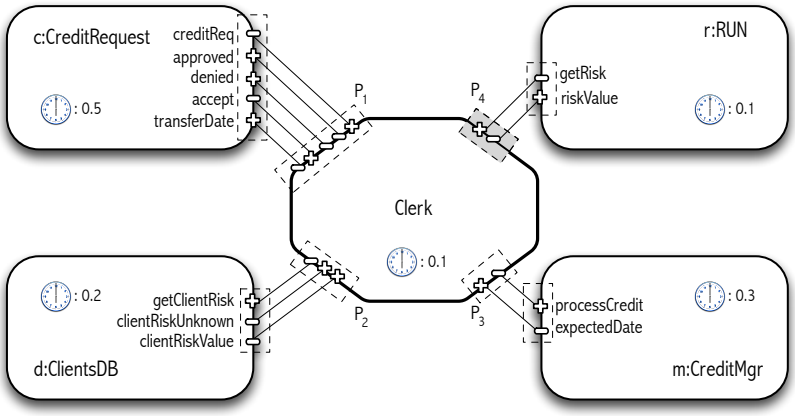

Figure 7: A HT-ARN consisting of a connection with a free port and three processes.

The alphabet of this HT-ARN is the set

$$\{c.\mathit{creditReq}!, c.\mathit{accept}!, d.\mathit{clientRiskValue}!, d.\mathit{clientRiskUnknown}!,$$
$$m.\mathit{expectedDate}!, r.\mathit{getRisk}!, c.\mathit{approved}_¡, c.\mathit{denied}_¡, c.\mathit{transferDate}_¡,$$

$d.getClientRisk¡, m.processCredit¡, r.riskValue¡\}$

To illustrate how the projections work, Fig. 8 depicts a timed trace of the HT-ARN (in the middle) and its projections to the alphabet of *Clerk* (above) and to the alphabet of *CreditRequest* (below) — for simplicity, we omitted the empty actions except where they appear as projections. For the middle trace to be accepted as a behaviour of the HT-ARN, its projections to *Clerk*, *CreditRequest*, RUN, *ClientsDB* and *CreditMgr* would have to be traces of the corresponding processes. This is the case of RUN because this is a process that accepts all traces. As can be seen in Figs. 4 and 5, the trace also projects to a behaviour of *Clerk* and a behaviour of *CreditRequest* (it refines the trace in Fig. 3). It is also easy to see that the projections to *ClientsDB* and *CreditMgr* satisfy the informal description of their behaviour as given in the previous section. In Sec. 6, we discuss a logic in which specifications of process behaviour can be formalised and against which traces can be formally checked.

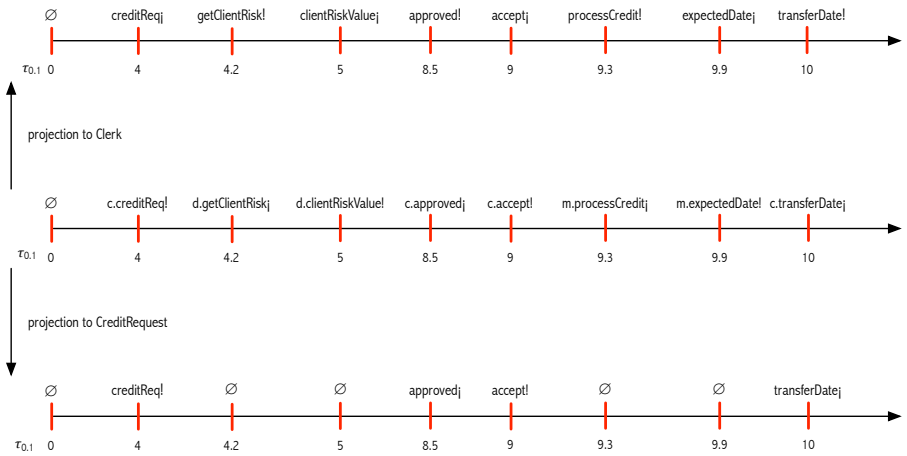

Figure 8: Projection of a trace of the HT-ARN in Fig. 7 (in the middle) to the alphabet of *Clerk* (above) and of *CreditRequest* (below).

An important property of HT-ARNs is that their behaviours are closed under refinement, which, as seen in the next section, makes them able to engage with other HT-ARNs.

Proposition 4.2. *For every* HT-*ARN* α, Λ_α *is r-closed.*

This follows from the fact that all processes are r-closed by construction, that translations preserve r-closure (Prop. 2.7) and that intersections of r-closed properties are r-closed (Prop. 2.5).

Although every HT-ARN is r-closed, not every HT-ARN has an 'equivalent' representation as a process, i.e., there is not necessarily a process that has the same alphabet of actions and the same set of traces as the HT-ARN, which would allow for the HT-ARN to be implemented over (or simulated by) a single processor. This is because, when time is heterogenous, there is not necessarily a time granularity over which all the traces of a HT-ARN can be generated.

However, if a HT-ARN is connected, i.e., every pair of processes is linked via a path of connections, a common divisor exists for all clock granularities.

In this case, although an equivalent process does not necessarily exist, a best approximation can be found in the sense of Def. 2.4:

Theorem 4.3. *If a* HT-*ARN* α *is connected and* $\Lambda_\alpha \neq \emptyset$*, there is a process* $P_\alpha = \langle \delta_\alpha, \gamma_\alpha, \Lambda_{P_\alpha} \rangle$ *such that* $\Lambda_{P_\alpha} \precapprox \Lambda_\alpha$ *and, for any other process* $P = \langle \delta, \gamma_\alpha, \Lambda_P \rangle$ *such that* $\Lambda_P \precapprox \Lambda_\alpha$*,* $\Lambda_P \precapprox \Lambda_{P_\alpha}$*.*

A constructive proof is given in the Appendix.

4.2. Composition

Two HT-ARNs can be composed through the ports that are still available for establishing further interconnections, i.e., not connected to any other port, which we call *interaction-points*.

Definition 4.4 (Interaction-point)**.** *An interaction-point of a* HT-*ARN* $\alpha = \langle N, E, \Omega, \Xi \rangle$ *is a pair* $\langle v, M \rangle$ *such that* $v \in N$ *is a node and* $M \in \gamma_v$ *is a port of the process* Ω_v *at that node, and either (1)* M *is not involved in any attachment of any hyperedge to which* v *belongs — what we call a* process interaction-point*, or (2)* Ω_v *is a* RUN *process* $\boxdot^\delta_M$ *and* v *belongs to an hyperedge* c *such that* $M^{op} \in \gamma_{cF}$ *(is a free port) — what we call a* connection interaction-point*. We denote by* J_α *the collection of interaction-points of* α*.*

That is, either M is not attached to any port (process interaction-point) or M is the port of a RUN process attached to a free port of a connection (connection interaction-point). For example, the HT-ARN depicted in Fig. 7 has only one interaction-point: $\langle r, \{getRisk, riskValue\} \rangle$, which is a connection interaction-point. Notice that, if $\langle v, M \rangle$ is a connection interaction point, there is only an hyperedge to which v belongs, which we denote by c_v.

We can interconnect two HT-ARNs by merging process interaction-points with connection interaction-points via attachments that are well-formed:

Definition 4.5 (Composition of HT-ARNs)**.** *Let* $\alpha = \langle N_\alpha, E_\alpha, \Omega_\alpha, \Xi_\alpha \rangle$ *and* $\beta = \langle N_\beta, E_\beta, \Omega_\beta, \Xi_\beta \rangle$ *be two* HT-*ARNs with disjoint sets of nodes and* θ *be a family of wires between* α *and* β*, where every wire is a triple*

$$\theta_i = \langle \langle v_i, M_{v_i} \rangle, \xi_i, \langle p_i, M_{p_i} \rangle \rangle$$

such that either

1. $\langle v_i, M_{v_i} \rangle$ *is a connection interaction-point of* α*,* $\langle p_i, M_{p_i} \rangle$ *is a process interaction-point of* β *and* $\langle \Omega_{\alpha c_{v_i}}, \xi_i : M^{op}_{v_i} \to M_{p_i}, \Omega_{\beta p_i} \rangle$ *is a well-formed attachment, or*

2. $\langle v_i, M_{v_i} \rangle$ *is a connection interaction-point of* β*,* $\langle p_i, M_{p_i} \rangle$ *is a process interaction-point of* α *and* $\langle \Omega_{\beta c_{v_i}}, \xi_i : M^{op}_{v_i} \to M_{p_i}, \Omega_{\alpha p_i} \rangle$ *is a well-formed attachment,*

and all sets of interaction-points are mutually disjoint, i.e., no interaction-point can be involved in more than one wire.
We define the HT-*ARN* $\alpha \parallel_\theta \beta = \langle N, E, \Omega, \Xi \rangle$ *as follows:*

- N *is obtained from* $N_\alpha \cup N_\beta$ *by removing the nodes corresponding to the connection interaction-points, and* E *is obtained from* $E_\alpha \cup E_\beta$ *by replacing, for each wire* θ_i*, the node* v_i *by* p_i *in the hyperedge* c_{v_i}*.*

- *Its node-labelling function* Ω *coincides with* Ω_α *or* Ω_β *on the corresponding remaining nodes.*
- *Its hyperedge-labelling function* Ξ *is as* $\Xi_\alpha \cup \Xi_\beta$ *except that, for each wire* θ_i*, the attachment of the run process at* v_i *is replaced with* ξ_i *and* $M^{op}_{v_i}$ *is removed from the set of free ports of the connection that labels the hyperedge* c_{v_i}*.*

In order to illustrate this operation, consider the HT-ARN depicted in Fig. 9, whose hypergraph has $\{e, b\}$ as its set of nodes and as its single hyperedge (that is, the hyperedge connects all nodes). The actual processes involved are not relevant for the composition so we omit them.

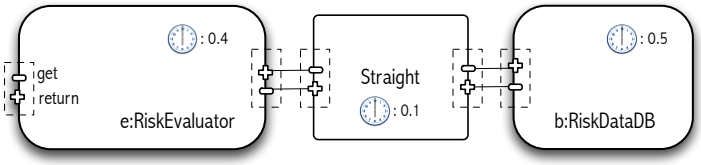

Figure 9: A HT-ARN that provides a risk-evaluation service

This HT-ARN has as a process interaction-point the pair $\langle e, \{get, return\}\rangle$, which we can use to compose with the HT-ARN depicted in Fig. 7 via the connection interaction point $\langle r, \{getRisk, riskValue\}\rangle$. The corresponding wire identifies *getRisk* with *get* and *riskValue* with *return*. The result of the composition is depicted in Fig. 10: an hypergraph with five nodes and two hyperedges, the node e belonging to both hyperedges.

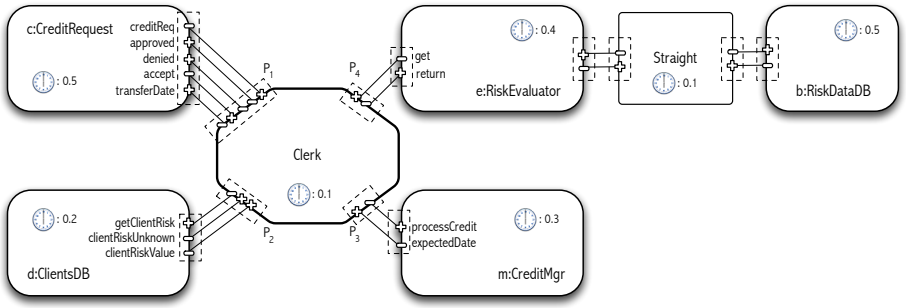

Figure 10: The composition of the HT-ARNs of Fig. 7 and Fig. 9.

Composition can take place at run time, allowing for parties to join connections at ports that are free, for example using service-oriented middleware.

5. Consistency

The joint consistency of the processes and the orchestrators operating in a HT-ARN is an important property because it ensures that their implementations can work together; consistency is understood as the existence of a timed trace of the HT-ARN that projects to traces of all the parties involved, i.e., one in which all processes and orchestrators agree.

Definition 5.1 (Consistent HT-ARN)**.** *A* HT-*ARN* α *is said to be* consistent *iff* $\Lambda_\alpha \neq \emptyset$.

For example, the HT-ARN depicted in Fig. 7 is consistent because, as argued in the previous section, the trace depicted in the middle of Fig. 8 projects to traces of all the processes and of the orchestrator.

5.1. Progress-enabled HT-ARNs

In general, identifying a joint trace that is accepted by all the parties (processes and orchestrators) is not easy; a proof of consistency is not normally constructive. Furthermore, consistency is an existential property that does not necessarily inform us of how the parties should behave to jointly build a timed trace. A related property is instead the ability of HT-ARNs to make progress, i.e., that any joint segment can be extended through a set of actions in which the parties (not necessarily all, or even any, given that time can just be allowed to flow) can engage in.

In [13] we characterised that property in an un-timed model, which we now extend to HT-ARNs by considering progress along a fixed time sequence.

Definition 5.2 (Progress-enabled)**.** *For every* HT-*ARN* $\alpha = \langle N, E, \Omega, \Xi \rangle$ *and time sequence* τ*, let*

$$\Pi_{\alpha_\tau} = \{\pi \in (2^{\mathsf{A}_\alpha})^* : \forall p \in N \cup E \ (\pi|_{\iota_p} \in \downarrow \Lambda_{p_\tau})\}$$

We say that α *is* progress-enabled in relation to τ *iff*

$$\epsilon \in \Pi_{\alpha_\tau} \ \textit{and} \ \ \forall \pi \in \Pi_{\alpha_\tau} \ \exists A \subseteq \mathsf{A}_\alpha((\pi \cdot A) \in \Pi_{\alpha_\tau})$$

We say that α *is* progress-enabled *iff there is a time sequence* τ *such that* α *is progress-enabled in relation to every* $\tau' \preceq \tau$.

The set Π_{α_τ} consists of all the segments that the processes can jointly engage in across the time sequence τ.

Notice that if $\epsilon \notin \Pi_{\alpha_\tau}$ then τ is not a refinement of δ_p-time sequence for at least one node or hyperedge p of α. Therefore, being progress enabled in relation to τ means that τ is an admissible time sequence for every node or hyperedge p of α and that, after any initial joint segment, all the processes can keep making progress along τ. Furthermore, because the intersection of A with the alphabet of any process can be empty, being progress-enabled does not require all parties to actually perform an action as long as being idle is admissible for those parties.

By itself, being progress-enabled does not guarantee that a HT-ARN is consistent: it guarantees the ability of a HT-ARN to make any number of finite moves but guaranteeing the existence of an infinite behaviour requires the analysis of what happens 'at the limit'. To do so, we need to use the topological properties of the space of timed traces.

Definition 5.3 (Closure relative to time)**.** *A t-closed* HT-*ARN is one in which all processes that label nodes or hyperedges (connections) are t-closed in the sense of Def. 2.3.*

Processes that are t-closed define safety properties in the usual un-timed sense: over a fixed time sequence, which cannot be controlled by the process, the violation of the property can be checked over a finite trace. In Sec. 6, we show how t-closure is related with the usual notion of safety for timed traces.

Corollary 5.4. *Let α be a* HT*-ARN. If α is t-closed then so is Λ_α.*

The corollary follows from the fact that the intersection of t-closed properties is also t-closed.

The following theorem generalises to a timed domain the characterisation of consistency that we proved in [13].

Theorem 5.5. *A* HT*-ARN is consistent if it is t-closed and progress-enabled.*

5.2. Compositionality

We now show how HT-ARNs can be guaranteed to be progress-enabled by construction: we identify atomic HT-ARNs that are progress-enabled and prove that the class of progress-enabled HT-ARNs is closed under composition. We start by remarking that, because processes are consistent and r-closed, given a process P, the HT-ARN that consists of a single node labelled with P is progress-enabled in relation to at least a δ-time sequence and all its refinements, and therefore is progress-enabled. The same applies to any HT-ARN that consists of a finite set of unconnected processes – in fact, this generalises to any finite juxtaposition of progressed-enabled HT-ARNs (or, indeed, consistent HT-ARNs); the challenge is in checking that progress-enabled HT-ARNs are closed under composition because composition connects HT-ARNs, i.e., it creates connected components.

In [13], we gave criteria for the composition of two (un-timed) progress-enabled ARNs to be progress-enabled based on the ability of processes to buffer incoming messages – being 'delivery-enabled' – and of connections to buffer published messages – being 'publication-enabled'. In a timed domain, it becomes necessary to identify time sequences across which all parties can work together. Given a HT-ARN and one of its interaction-points $\langle v, M\rangle$, we define the set $D_{\langle v,M\rangle}$ of deliveries that can be made at that point:

- $D_{\langle v,M\rangle}=\{v.m¡ : m{\in}M^+\}$ if $\langle v, M\rangle$ is a process interaction-point.
- $D_{\langle v,M\rangle} = \{v.m! : m{\in}M^-\}$ if $\langle v, M\rangle$ is a connection interaction-point.

Notice that in the latter case we are actually interested in the deliveries that are made to the orchestrator, which are the publications made at the free port.

Definition 5.6 (Delivery-enabled HT-ARN)**.** *A* HT*-ARN α is delivery-enabled in relation to one of its interaction-points $\langle v, M\rangle{\in}J_\alpha$ if, for every $B \subseteq D_{\langle v,M\rangle}$, $\tau{\in}\Lambda_{time}$ and $(\pi{\cdot}A){\in}\Pi_{\alpha_\tau}$ such that $\tau(|\pi|)$ is a multiple of the time granularity δ_v of the process at v (i.e., α makes a step in sync with the process at v), $(\pi \cdot B \cup (A \setminus D_{\langle v,M\rangle})) \in \Pi_{\alpha_\tau}$ (i.e., α accepts the actions in B instead of those in A).*

That is, being delivery-enabled at an interaction point requires any joint segment of the HT-ARN over a time sequence to be extensible with any set of messages delivered at that interaction-point. Note that in the case of a connection interaction-point, being delivery-enabled means that the orchestrator of the connection is ready to accept publications made at the node v (the free port). Also note that being delivery-enabled does not interfere with the decision to publish messages: $B \cup (A{\setminus}D_{\langle v,M\rangle})$ retains all the publications in A. See also Fig.11.

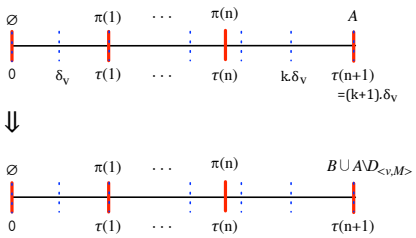

Figure 11: Being delivery enabled: deliveries are accepted at any multiple of the granularity of the interaction point.

Finally, we need to make sure that the processes that orchestrate connections can work together with the processes that they interconnect, i.e., that they do not force the delivery of messages when the processes cannot receive them:

Definition 5.7 (Cooperative process)**.** *Let $C = \langle \delta, \gamma, \Lambda \rangle$ be a process. For every port $M \in \gamma$, let $E_M = \{m! : m \in M^-\}$. The process is said to be* cooperative *in relation to M and a multiple δ' of δ if, for every $(\pi \cdot A) \in \downarrow\Lambda_{\tau_\delta}$, if $\tau(|\pi|)$ is not a multiple of δ' then $\pi \cdot (A \setminus E_M) \in \downarrow\Lambda_{\tau_\delta}$.*

That is, if after π the process wants to publish at a port M when it is not in sync with δ', there is an alternative path from π where no publication is made at that time. Notice that $\tau(|\pi|)$ is the time at which A is executed. See also Fig.12.

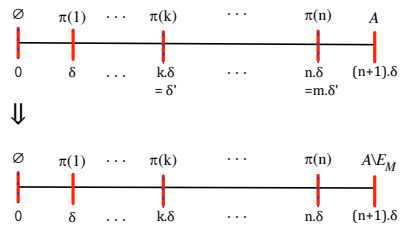

Figure 12: Being cooperative: deliveries to peers are not forced when they are not in sync.

We can now state our main compositionality result:

Theorem 5.8. *Let α be a composition of progress-enabled* HT*-ARNs through a family of wires with mutually-disjoint sets of interaction points i.e.,*

$$\alpha = (\alpha_1 \; ||^{i=1\ldots n}_{\langle\langle v_i, M_{v_i}\rangle, \xi_i, \langle p_i, M_{p_i}\rangle\rangle} \; \alpha_2)$$

where each $\langle\langle v_i, M_{v_i}\rangle, \xi_i, \langle p_i, M_{p_i}\rangle\rangle$ is a wire between α_1 and α_2. If the orchestrators involved in the wires (those that label the hyperedges c_{v_i}) are cooperative in relation to the free ports M_{v_i} that are being connected and the time granularity of the processes at the corresponding interaction points $\langle p_i, M_{p_i}\rangle$, and both HT*-ARNs are delivery-enabled in relation to the interaction-points being connected, then α is progress-enabled.*

To further guarantee that the HT-ARN that results from the composition is consistent, it is sufficient to choose processes and orchestrators that are t-closed

(implement safety properties), which is something that can be done at design time, not at composition time.

Another important property is that composition preserves being delivery-enabled in the following sense:

Proposition 5.9. *Let α be the composition of two* HT-*ARNs α_1 and α_2 through a family of wires. Let $\langle v, M\rangle$ be an interaction-point of one of the α_i that is not involved in any of the wires. If α_i is delivery-enabled in relation to $\langle v, M\rangle$, so is α.*

Because every HT-ARN can be seen as the result of a composition of elementary networks (individual processes or connections), this means that the proof that an HT-ARN is delivery-enabled can be reduced to checking that individual processes (including the orchestrators) are delivery-enabled in relation to their ports. That is, the checking can be done at design time, not at composition time. Checking that an individual process is delivery-enabled can be done as follows:

Corollary 5.10 (Delivery-enabled process)**.** *A process $\langle \delta, \gamma, \Lambda\rangle$ is delivery-enabled in relation to one of its ports M if, for every $B \subseteq D_M = \{m_i : m \in M^+\}$ and $(\pi{\cdot}A) \in {\downarrow}\Lambda_{\tau_\delta}$, $(\pi \cdot B \cup (A \setminus D_M)) \in {\downarrow}\Lambda_{\tau_\delta}$.*

Checking that processes are cooperative in relation to their free ports and multiples of their clock granularities can also be done at design time though the multiples that are of interest will only be known at composition time.

5.3. An automata-theoretic view

Checking that processes are delivery-enabled and that connections are cooperative is easier over more operational abstractions of the implementations that generate behaviours, such as automata. In this section, we give as an example abstractions of implementations of processes through discrete timed I/O machines as defined in [11].

We start by recalling the notion of timed I/O automata (TIOA) as defined in [9] except that transitions perform sets of actions instead of single actions. A TIOA is defined in terms of a finite set $\mathbb{C}$ of clocks. A *condition* over $\mathbb{C}$ is a finite conjunction of expressions of the form $c \bowtie n$ where $c \in \mathbb{C}$, $\bowtie \in \{\leq, \geq\}$ and $n \in \mathbb{N}$. We denote by $\mathcal{B}(\mathbb{C})$ the set of conditions over $\mathbb{C}$.

Definition 5.11 (TIOA)**.** *A timed I/O automaton $\mathcal{A}$ (TIOA) is a tuple*

$$\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv\rangle$$

where:

- *Loc is a finite set of locations;*
- *$q_0 \in Loc$ is the initial location;*
- *$\mathbb{C}$ is a finite set of clocks;*
- *$E \subseteq Loc \times 2^{Act} \times \mathcal{B}(\mathbb{C}) \times 2^{\mathbb{C}} \times Loc$ is a finite set of edges;*
- *$Act = Act^I \cup Act^O \cup Act^\tau$ is a finite set of actions partitioned into inputs, outputs and internal actions, respectively;*

- *Inv: $Loc \to \mathcal{B}(\mathbb{C})$ is a mapping that associates an invariant with every location.*

In addition, we impose that every TIOA does not interfere with the ability of the environment to make progress, i.e., is open*: for all $l \in Loc$, there is an edge $(l, \emptyset, \phi, \emptyset, l') \in E$ for some location l' such that $Inv(l')$ is implied by $Inv(l)$ and for some tautology ϕ.*

Given an edge (l, S, C, R, l'), l is the source location, l' is the target location, S is the set of actions executed during the transition, C is a guard (a condition that determines if the transition can be performed), and R is the set of clocks that are reset by the transition. The requirement that every location is the source of a transition labelled by $\emptyset$ that is always enabled means that the behavior of $\mathcal{A}$ is always open to the execution of actions in which it is not involved.

A *clock valuation* over a set $\mathbb{C}$ of clocks is a mapping v: $\mathbb{C} \to \mathbb{R}_{\geq 0}$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, we denote by $v+d$ the valuation defined by, for any clock $c \in \mathbb{C}$, $(v+d)(c) = v(c)+d$. Given $R \subseteq \mathbb{C}$ and a clock valuation v, we denote by $v^{\mathbf{R}}$ the valuation where clocks from R are reset, i.e., such that $v^{\mathbf{R}}(c)=0$ if $c \in R$ and $v^{\mathbf{R}}(c)=v(c)$ otherwise. Given a condition C in $\mathcal{B}(\mathbb{C})$, we use $v \vDash C$ to express that C holds for the clock valuation v.

Definition 5.12 (Execution)**.** *Let $\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$ be a TIOA. An* execution *of $\mathcal{A}$ starting in l_0 and valuation v_0 is a sequence*

$$(l_0, v_0, d_0) \overset{S_0,R_0}{\longrightarrow} (l_1, v_1, d_1) \overset{S_1,R_1}{\longrightarrow} \dots$$

where, for all i:

(1) $l_i \in Loc$, v_i is a clock valuation over $\mathbb{C}$ and $d_i \in \mathbb{R}_{>0}$;

(2) $S_i \subseteq Act$ and $R_i \subseteq \mathbb{C}$;

(3) for all $0 \leq t \leq d_i$, $v_i + t \vDash Inv(l_i)$;

(4) $v_{i+1}=(v_i + d_i)^{\mathbf{R_i}}$; and

(5) there is $(l_i, S_i, C, R_i, l_{i+1}) \in E$ such that $v_i + d_i \vDash C$.

A partial execution *is of the form*

$$(l_0, v_0, d_0) \overset{S_0,R_0}{\longrightarrow} \dots \overset{S_{n-1},R_{n-1}}{\longrightarrow} (l_n, v_n, d_n)$$

where (1) and (3) hold for all $i \in [0..n]$, and (2), (4) and (5) for all $i \in [0..n-1]$.

That is, each triple (l_i, v_i, d_i) consists of a location, the value of the clocks when that location is reached at that point of the execution, and the duration for which the automaton remains at that location before the next transition (which can leave the automaton in the same location). During this time, the invariant $Inv(l_i)$ must hold. A transition out of (l_i, v_i, d_i) happens at the end of d_i units of time and needs to be made by an edge whose guard C_i holds at that time and leads to a location whose invariant is satisfied. As a result of the transition, the clocks are updated to $(v_i + d_i)^{\mathbf{R_i}}$.

A pair (l, v) where l is a location and v is a clock valuation is said to be *reachable at time* $T \in \mathbb{R}_{\geq 0}$ if either (a) $(l, v) = (q_0, 0)$, $T = 0$ and, there exists $d_0 > 0$ such that $t \vDash Inv(q_0)$ for all $0 \leq t \leq d_0$; or (b) there exists a partial execution that starts at $(q_0, 0)$ and ends at $(l_n, v_n) = (l, v)$, and $T = \sum_{i=0..n-1} d_i$. Note that, in the last case, condition (3) of Def. 5.12 must hold until n and, hence, there exists $d_n > 0$ such that $Inv(l_n)$ holds for d_n time instants.

A timed machine is a TIOA that executes in the context of a clock granularity δ, i.e., its actions are always executed at instant times that are multiples of δ.

Definition 5.13 (DTIOM)**.** *A* discrete timed I/O machine *is a pair* $\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$ *where* $\delta_{\mathcal{M}} \in \mathbb{R}_{>0}$ *and* $\mathcal{A}_{\mathcal{M}}$ *is a TIOA.*

The executions *and* partial executions *of* $\mathcal{M}$ *are those of* $\mathcal{A}_{\mathcal{M}}$ *restricted to transitions at every* $\delta_{\mathcal{M}}$*, i.e.,*

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} (l_1, v_1, d_1) \xrightarrow{S_1, R_1} \ldots$$

such that all the durations d_i *are* $\delta_{\mathcal{M}}$*. Therefore, we represent executions of DTIOMs as sequences*

$$(l_0, v_0) \xrightarrow{S_0, R_0} (l_1, v_1) \xrightarrow{S_1, R_1} \ldots$$

and call each pair (l_i, v_i) *an execution state.*

The behaviour $[\![\mathcal{M}]\!]$ *of* $\mathcal{M}$ *is the set of executions such that* $l_0 = q_0$ *and* $v_0(c) = 0$ *for all* $c \in \mathbb{C}$*, i.e., those that start in the initial location with all clocks set to* 0*.*

Every execution of a DTIOM $\mathcal{M}$ *defines the* $\delta_{\mathcal{M}}$*-timed trace* $\lambda = \langle \sigma, \tau_{\delta_{\mathcal{M}}} \rangle$ *over Act where* $\sigma(0) = \emptyset$ *and, for* $i \geq 0$, $\sigma(i+1) = S_i$*. We denote by* $\Lambda_{\mathcal{M}}$ *the r-closure of the set of timed traces defined by* $[\![\mathcal{M}]\!]$*, which we call its* language*.*

Definition 5.14 (Implementation of a process defined by a DTIOM)**.** *An implementation of a process* $\langle \delta, \gamma, \Lambda \rangle$ *is a DTIOM* $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$ *such that:*

- $Act^I_{\mathcal{A}} = \bigcup_{M^+ \in \gamma} \mathsf{A}_{M^+}$
- $Act^O_{\mathcal{A}} = \bigcup_{M^- \in \gamma} \mathsf{A}_{M^-}$
- $Act^\tau_{\mathcal{A}} = \emptyset$
- $\Lambda = \Lambda_{\mathcal{M}}$

That is, the input actions of the DTIOM are the message deliveries, its outputs are the publications, and there are no internal actions (internal actions arise when DTIOM are composed). The behaviour of the process must be the language of the DTIOM.

Definition 5.15 (Cooperative)**.** *A DTIOM* $\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$*, where* $\mathcal{A}_{\mathcal{M}} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$*, is said to be* cooperative in relation to $Q \subseteq Act$ *and a multiple* δ *of* $\delta_{\mathcal{M}}$ *if the following holds for every* (l, v) *reachable at a time* T *such that* $(T + \delta_{\mathcal{M}})$ *is not a multiple of* δ*:*

> *for every edge* $(l, A, C, R, l') \in E$ *such that* $v + \delta_{\mathcal{M}} \vDash C$ *and* $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv(l')$ *for all* $0 \leq t \leq \delta_{\mathcal{M}}$ *— i.e., the machine makes a transition at a time that is not a multiple of* δ *— there exists an edge* $(l, A \backslash Q, C', R', l'')$

such that $v+\delta_{\mathcal{M}} \vDash C'$ *and for all* $0 \leq t \leq \delta_{\mathcal{M}}$, $(v+\delta_{\mathcal{M}})^{\mathbf{R}'}+t \vDash Inv(l'')$ — *i.e., the machine can make an alternative transition that does not perform any actions in* Q.

Essentially, being cooperative in relation to Q and δ means that the machine will not force transitions that perform actions in Q at times that are not multiples of δ.

The following result follows trivially from Def. 5.7:

Proposition 5.16. *Let* P *be a process with granularity* δ, M *one of its ports and* $\mathcal{M}$ *one of its implementations. If* $\mathcal{M}$ *is cooperative in relation to* A_{M^-} *and a multiple* δ' *of* δ, *then* P *is cooperative in relation to* M *and* δ'.

Definition 5.17 (DP-enabled)**.** *A DTIOM* $\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$, *where* $\mathcal{A}_{\mathcal{M}} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$, *is* DP-enabled in relation to $J \subseteq Act^I$ *if the following property holds for every* $B \subseteq J$ *and state* (l, v) *reachable at a time* T*:*

> *for every edge* $(l, A, C, R, l') \in E$ *such that* $v + \delta_{\mathcal{M}} \vDash C$ *and, for all* $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv(l')$ — *i.e., the machine can make a transition — there exists an edge* $(l, B \cup (A \backslash J), C', R', l'')$ *such that* $v + \delta_{\mathcal{M}} \vDash C'$ *and, for all* $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv(l'')$ — *i.e., the machine can make an alternative transition that accepts instead* B *as inputs and still performs the same outputs (and inputs outside* J*).*

The term DP-enabled stands for delivery/publication-enabled and it reflects that inputs received by machines in a network with the role of components are deliveries made by orchestrators, and inputs received by orchestrators are publications made by components.

Therefore, a DTIOM is DP-enabled in relation to a set of inputs J if, whenever it leaves a reachable state, it can do so by accepting any subset of J, and if its outputs are independent of the inputs in J that it receives.

The following result follows trivially from Cor. 5.10:

Proposition 5.18. *Let* P *be a process with granularity* δ, M *one of its ports and* $\mathcal{M}$ *one of its implementations. If* $\mathcal{M}$ *is DP-enabled in relation to* A_{M^+}, *then* P *is delivery-enabled in relation to* M.

In order to estimate the complexity of checking these properties, it is useful to use the Büchi-automata "equivalent" of DTIOM defined in [11].

Let $\mathcal{A} = \langle Loc, l_0, \mathbb{C}, E, Act, Inv \rangle$ be a TIOA. Given a clock c, let $\mathsf{Max}^{\mathcal{A}}(c)$ denote the maximal constant with which c is compared in the guards and invariants of $\mathcal{A}$. Let $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$ and $\mathcal{B}_{\mathcal{M}} = \langle Q, q_0, 2^{Act}, \rightarrow, Q \rangle$ be the Büchi automaton such that:

- $Q = Loc \times \prod_{c \in \mathbb{C}} [0 \,..\, \lfloor \frac{\mathsf{Max}^{\mathcal{M}}(c)}{\delta} \rfloor + 1]$ (i.e., states consist of a location l and a natural number $n_c \leq \lfloor \frac{\mathsf{Max}^{\mathcal{M}}(c)}{\delta} \rfloor + 1$, for every $c \in \mathbb{C}$);
- $q_0 = (l_0, \mathbf{0})$;
- $(l, \boldsymbol{\nu}) \xrightarrow{S} (l', \boldsymbol{\nu}')$ iff there exists a transition $(l, S, C, R, l') \in E$ such that:
  - (i) for all $0 \leq t \leq \delta$, $\boldsymbol{\nu} \cdot \delta + t \models Inv(l)$,
  - (ii) $\boldsymbol{\nu} \cdot \delta + \delta \models C$,

(iii) for all $c \in \mathbb{C}$, $\boldsymbol{\nu}'(c) = \left\{ \begin{array}{ll} 0 & \text{if } c \in R \\ \boldsymbol{\nu}(c) & \text{if } c \notin R \text{ and } \boldsymbol{\nu}(c) = \lfloor \frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta} \rfloor + 1 \\ \boldsymbol{\nu}(c) + 1 & \text{otherwise} \end{array} \right\}$

(iv) $\boldsymbol{\nu}' \cdot \delta \models Inv(l')$.

Notice that Q involves only natural numbers. The size of $\mathcal{B}_{\mathcal{M}}$ is in $O(|Loc| \cdot (\lfloor \frac{\mathsf{Max}}{\delta} \rfloor + 2)^{|\mathbb{C}|})$, where $|Loc|$ and $|\mathbb{C}|$ are the size of Loc and the number of clocks, respectively, and $\mathsf{Max} = \max\{\mathsf{Max}^{\mathcal{A}}(c) \mid c \in \mathbb{C}\}$ is the maximal constant considered in all constraints and invariants of $\mathcal{M}$.

The Büchi automaton $\mathcal{B}_{\mathcal{M}}$ is equivalent to $\mathcal{M}$ in the following sense:

Theorem 5.19 ([11]). *For all action sequences σ over Act, $\langle \sigma, \tau_\delta \rangle \in [\![\mathcal{M}]\!]$ iff the infinite sequence $\sigma(1)\sigma(2)\ldots$ is in the language of $\mathcal{B}_{\mathcal{M}}$.*

In practice, being cooperative can be verified using a syntactic check on the states of the equivalent Büchi automaton that can be reached with a number of transitions n such that $n+1$ is not a multiple of $\delta/\delta_{\mathcal{M}}$. This can be done in time in $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2)$, with $|\mathcal{B}_{\mathcal{M}}|$ being the size of the Büchi automaton $\mathcal{B}_{\mathcal{M}}$.

Likewise, DP-enabledness can be verified in $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2 \cdot 2^{|Act^I_{\mathcal{M}}|})$.

6. A Compositional Theory for HT-ARNs

In this section, we discuss a logic that supports the specification of timed properties as defined in Sec. 2, and define a specification theory for our component algebra. More specifically, we want to be able to specify processes through a finite collection Φ of sentences of the logic and use the inference mechanisms of the logic to derive properties of processes and HT-ARNs using the specifications of the intervening processes. This requires a logic in which we can specify and reason about timed traces.

We are interested in answering the following specific questions:

δ-Satisfiability Given a specification Φ and a clock granularity δ, is there a process $\langle \delta, \gamma, \Lambda \rangle$ that satisfies Φ?

δ-Canonicity Given a specification Φ and a clock granularity δ, is there a canonical process (in the sense of being maximal for Φ and δ) that satisfies Φ?

The typical answer to these questions is, for satisfiability, that the specification is consistent (i.e., $\Lambda_\Phi = \{\lambda : \lambda \vDash \Phi\} \neq \emptyset$), and for canonicity that one takes Λ_Φ as the canonical model of Φ (the largest set of timed traces that satisfy the specification).

Because the behaviour of a process is the r-closure of a non-empty δ-timed property (Def. 3.1), where δ is a clock granularity, and that we are interested in t-closed processes in order to guarantee good properties of HT-ARNs (such as consistency), the answers to our questions are not so simple and lead to another set of questions:

- Is there a logic in which we can guarantee that Λ_Φ is t-closed, r-closed, and such that all its models are refinements of a δ-timed trace?

- If not, is there an alternative way of defining a canonical model?
- If $\Lambda_\Phi \neq \emptyset$, i.e., if there is a timed trace that satisfies Φ, can we guarantee satisfiability, i.e., the existence of a process that satisfies Φ?

More specific questions that would help us choose an appropriate logic are:

1. Is the logic closed under limits in the Cantor topology for a fixed time sequence, i.e., if a set of timed traces satisfies Φ, can we guarantee that its t-closure also satisfies Φ?
2. Is the logic closed under refinement, i.e., if a set of timed traces satisfies Φ, can we guarantee that its r-closure also satisfies Φ?

If the answers are positive, then we could choose as a behaviour for the canonical process the set Λ^δ_Φ that is the r-closure of the t-closure of the set $\{\lambda : \lambda \in \mathsf{ttra}_\delta(\mathsf{A}_\gamma) \wedge \lambda \vDash \Phi\}$, i.e., we choose first the δ-timed traces that satisfy Φ, and then close it for time, and then for refinement. This would give us the r-closure of a δ-timed property that is also t-closed, which would satisfy Φ. Sec. 6.1 and Sec. 6.2 discuss answers to those two questions.

In order to check for δ-satisfiability, in particular that the set Λ^δ_Φ is not empty (and, hence, defines a process), one would have to find a δ-timed trace (or the refinement of a δ-timed trace) that satisfies Φ. This could be done, for example, by building a DTIOM (which by definition generates only δ-timed traces) that satisfies Φ. An alternative, which we explore in Sec. 6.3, is to identify a formula $\mathbb{A}x_\delta$ that is only satisfied by timed traces that are refinements of a δ-timed trace. In this case, one would check for the satisfiability of Φ and $\mathbb{A}x_\delta$ over the space of all timed traces.

6.1. Logics for specifying timed properties

Several extensions of LTL have been proposed to express and reason about real time, which are usually based on Metric Temporal Logic (MTL) [24]. MTL works over timed traces and has been studied extensively in relation to important properties such as satisfiability and model-ckecking. The formulas of MTL are built from a set of A of actions (atomic propositions) using Boolean connectives and time-constrained versions of the until operator of the form $\mathcal{U}_I$ where $I \subseteq [0, \infty)$ is an interval with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$:

$$\phi ::= a \mid \neg\phi \mid \phi \supset \phi \mid \phi \,\mathcal{U}_I\, \phi$$

where $a \in \mathsf{A}$. Fragments of MTL have been characterised in which only safety properties can be expressed; for example, SAFETY-MTL [26] requires that sentences are in negation normal form and all eventualities are time-bounded. SAFETY-MTL has a pointwise semantics, i.e., one evaluates sentences over the indexes of timed traces.

Definition 6.1 (Pointwise semantics of MTL)**.** *For every timed trace $\lambda = \langle \sigma, \tau \rangle$ over A and $i \in \mathbb{N}$:*

- $\lambda, i \vDash_{pw} a$ *iff* $a \in \sigma(i)$
- $\lambda, i \vDash_{pw} \neg\phi$ *iff* $\lambda, i \nvDash_{pw} \phi$
- $\lambda, i \vDash_{pw} \phi_1 \supset \phi_2$ *iff if* $\lambda, i \vDash_{pw} \phi_1$ *then* $\lambda, i \vDash_{pw} \phi_2$

- $\lambda, i \vDash_{pw} \phi_1 \ \mathcal{U}_I \ \phi_2$ *iff there exists* $k \geq i$ *such that* $(\tau(k) - \tau(i)) \in I$, $\lambda, k \vDash_{pw} \phi_2$ *and, for all* $i \leq j < k$, $\lambda, j \vDash_{pw} \phi_1$

We say that λ *satisfies* ϕ *in the pointwise semantics, denoted by* $\lambda \vDash_{pw} \phi$, *iff* $\lambda, 0 \vDash_{pw} \phi$. *We write* $\Lambda \vDash_{pw} \phi$ *to mean that, for all* $\lambda \in \Lambda$, $\lambda \vDash_{pw} \phi$; $\lambda \vDash_{pw} \Phi$ *to mean that, for all* $\phi \in \Phi$, $\lambda \vDash_{pw} \phi$; *and* $\Lambda \vDash_{pw} \Phi$ *to mean that, for all* $\phi \in \Phi$, $\Lambda \vDash_{pw} \phi$.

In a time context, a safety property Λ is one that is *divergent safe* [20], i.e., for any timed trace λ, if for all $\pi{<}\lambda$ there is $\lambda'{\in}\Lambda$ such that $\pi{<}\lambda'$, then $\lambda{\in}\Lambda$.

Proposition 6.2. *Given a set* Φ *of sentences in* SAFETY-*MTL,* $\Lambda_\Phi^{pw} = \{\lambda : \lambda \vDash_{pw} \Phi\}$ *is divergent safe.*

It is easy to see that divergent-safe properties are also t-closed:

Proposition 6.3. *Let* Λ *be a timed property. If* Λ *is divergent safe then it is also t-closed.*

Corollary 6.4. *Let* Φ *be a set of sentences in* SAFETY-*MTL. Then,*

- Λ_Φ^{pw} *is t-closed.*
- *If* $\Lambda \vDash_{pw} \Phi$ *then* $\Lambda^t \vDash_{pw} \Phi$.

This answers Question 1 above. Furthermore, SAFETY-MTL is fully decidable [26]. However, Λ_Φ^{pw} is not necessarily r-closed (even when Φ is in SAFETY-MTL) and, more generally, if $\Lambda \vDash_{pw} \Phi$ then it is not necessarily the case that $\Lambda^r \vDash_{pw} \Phi$, even if Λ is divergent safe. This is why in [14] we adopted a continuous semantics defined in terms of signals [21]. Herein, we use instead the continuous interpretation over timed traces proposed in [30], which is equivalent but simpler.

Definition 6.5 (Continuous semantics of MTL)**.** *For every timed trace* $\lambda = \langle \sigma, \tau \rangle$ *over* A *and* $t \in \mathbb{R}_{\geq 0}$*:*

- $\lambda, t \vDash_c a$ *iff exists* $i \in \mathbb{N}$ *such that* $\tau(i) = t$ *and* $a \in \sigma(i)$
- $\lambda, t \vDash_c \neg\phi$ *iff* $\lambda, t \nvDash_c \phi$
- $\lambda, t \vDash_c \phi_1 \supset \phi_2$ *iff if* $\lambda, t \vDash_c \phi_1$ *then* $\lambda, t \vDash_c \phi_2$
- $\lambda, t \vDash_c \phi_1 \ \mathcal{U}_I \ \phi_2$ *iff there exists* $u \geq t$ *such that* $(u - t) \in I$, $\lambda, u \vDash_c \phi_2$ *and, for all* $t \leq r < u$, $\lambda, r \vDash_c \phi_1$

We say that λ *satisfies* ϕ *in the continuous semantics, denoted by* $\lambda \vDash_c \phi$, *iff* $\lambda, 0 \vDash_c \phi$. *We write* $\Lambda \vDash_c \phi$ *to denote that, for all* $\lambda \in \Lambda$, $\lambda \vDash_c \phi$.

According to the terminology of [18], this version of MTL is non-strict and non-matching: $\phi_1 \ \mathcal{U} \ \phi_2$ does not constrain strictly the future or the current time, and does not require ϕ_1 to hold together with ϕ_2. This variant, under an interpretation with dense time (based on signals), is proved in [18] to be equivalent to the (most common) variant in which the until operator is strict and non-matching.

An important result is that the interpretation of a formula over a trace is the same over any of its refinements:

Proposition 6.6. *Given timed traces λ and λ' such that $\lambda' \preceq \lambda$, $\lambda \vDash_c \phi$ iff $\lambda' \vDash_c \phi$. It follows that, for every Φ, $\Lambda^c_\Phi = \{\lambda : \lambda \vDash_c \Phi\}$ is r-closed, and that, for every Λ, if $\Lambda \vDash_c \Phi$ then $\Lambda^r \vDash_c \Phi$.*

This answers Question 2 above. However, Λ^c_Φ is not necessarily t-closed and, more generally, $\Lambda \vDash_c \Phi$ does not necessarily imply that $\Lambda^t \vDash_c \Phi$. Furthermore, MTL has been proved to be undecidable in the continuous semantics based on signals [3].

Therefore, to be able to answer both Question 1 and Question 2, we decided to investigate the existence of a sub-logic of SAFETY-MTL that has the same semantics under a continuous and a pointwise semantics.

6.2. SAFETY-*MTL(R)*

We identify a fragment of SAFETY-MTL with the same continuous and pointwise semantics by making use of two functions that connect the continuous and the pointwise time domains:

Definition 6.7 (next, previous)**.** *Given a time sequence τ, we define the functions $next_\tau : \mathbb{R}_{\geq 0} \to \mathbb{N}$ and $previous_\tau : \mathbb{R}_{\geq 0} \to \mathbb{N}$ as follows:*

(a) $next_\tau(t) = \min\{i \in \mathbb{N} : \tau(i) \geq t\}$;

(b) $previous_\tau(t) = \max\{i \in \mathbb{N} : t \geq \tau(i)\}$.

For simplicity, given a timed trace $\lambda = \langle \sigma, \tau \rangle$, we use $next_\lambda$ and $previous_\lambda$ to mean $next_\tau$ and $previous_\tau$, respectively.

These functions, as shown in Fig. 13, map time instances to indexes of a given timed trace. It is easy to see that both functions map the time instants $\tau(i)$ into i. We use these functions to identify the formulas of SAFETY-MTL that, in the continuous semantics, are satisfied by all the timed traces that satisfy them under the pointwise semantics.

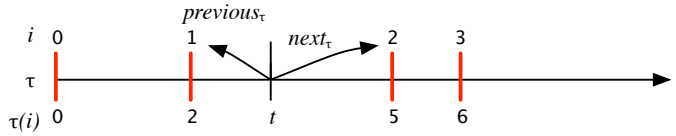

Figure 13: Next and previous functions.

Definition 6.8 (PW2C)**.** *Consider the sub-logic PW2C with the following syntax:*

$$\phi^\circ ::= true \mid false \mid a \mid \neg a \mid \phi^\circ \wedge \phi^\circ \mid \phi^\circ \vee \phi^\circ \mid \phi^\circ \; \mathcal{R}_{I^<} \; \phi^-$$
$$\mid \phi^\circ \; \mathcal{R}_{I^+} \; \phi^+ \mid \phi^\circ \; \mathcal{R}_{I^\forall} \; \phi^\diamond \mid \phi^- \; \mathcal{U}_I \; \phi^\circ$$
$$\phi^- ::= true \mid false \mid \neg a \mid \phi^- \wedge \phi^- \mid \phi^- \vee \phi^- \mid \phi^\circ \; \mathcal{R} \; \phi^- \mid \phi^- \; \mathcal{U}_{I_0^-} \; \phi^\circ$$
$$\phi^+ ::= true \mid false \mid \neg a \mid \phi^+ \wedge \phi^+ \mid \phi^+ \vee \phi^+ \mid \phi^\circ \; \mathcal{R} \; \phi^+$$
$$\phi^\diamond ::= true \mid \neg a \mid \phi^\diamond \wedge \phi^\diamond \mid \phi^\diamond \vee \phi^\diamond \mid \phi^\circ \vee \phi^\diamond$$

where $I^<$ is of the form $[0,t)$ or $[0,t]$ or $[0,\infty)$, I_0^- is of the form $(0,t)$ or $(0,t]$, I^+ is of the form $[t,\infty)$ or (t,∞), $I^\forall$ is any interval, and I is any bounded interval.

The pointwise and continuous semantics of formulas written in terms of *true*, *false*, $\wedge$, $\vee$, $\mathcal{R}$ are derived from Defs. 6.1 and 6.5, respectively, considering the usual abbreviations. In particular, the semantics of formulas of the form $\phi_1 \mathcal{R} \phi_2$ and $\phi_1 \mathcal{R}_I \phi_2$ is as follows:

- $\lambda, i \vDash_{pw} \phi_1 \mathcal{R} \phi_2$ iff for all $k \geq i$, $\lambda, k \vDash_{pw} \phi_2$ or there exists $i \leq j < k$ such that $\lambda, j \vDash_{pw} \phi_1$
- $\lambda, i \vDash_{pw} \phi_1 \mathcal{R}_I \phi_2$ iff for all $k \geq i$ such that $(\tau(k) - \tau(i)) \in I$, $\lambda, k \vDash_{pw} \phi_2$ or there exists $i \leq j < k$ such that $\lambda, j \vDash_{pw} \phi_1$

- $\lambda, t \vDash_c \phi_1 \mathcal{R} \phi_2$ iff for all $u \geq t$, $\lambda, u \vDash_c \phi_2$ or there exists $t \leq r < u$ such that $\lambda, r \vDash_c \phi_1$
- $\lambda, t \vDash_c \phi_1 \mathcal{R}_I \phi_2$ iff for all $u \geq t$ such that $(u - t) \in I$, $\lambda, u \vDash_c \phi_2$ or there exists $t \leq r < u$ such that $\lambda, r \vDash_c \phi_1$

The following properties of these four classes of formulas are of special interest:

Lemma 6.9. *Let $\lambda = \langle \sigma, \tau \rangle$ be a timed trace, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{\geq 0}$.*

1. *If for all $i \in \mathbb{N}$ $\tau(i) \neq t$ then $\lambda, t \vDash_c \phi^\diamond$*
2. *If $\lambda, next_\lambda(t) \vDash_{pw} \phi^+$ then $\lambda, t \vDash_c \phi^+$.*
3. *If $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ then $\lambda, t \vDash_c \phi^-$.*
4. *If $\lambda, i \vDash_{pw} \phi^\circ$ then $\lambda, \tau(i) \vDash_c \phi^\circ$.*

That is:

1. Formulas of the form $\phi^\diamond$ are satisfied in the continuous semantics at times that are not in the pointwise semantics.
2. Formulas of the form ϕ^- are satisfied in the continuous semantics at a time t if they are satisfied in the pointwise semantics at $previous_\lambda(t)$.
3. Formulas of the form ϕ^+ are satisfied in the continuous semantics at a time t if they are satisfied in the pointwise semantics at $next_\lambda(t)$.
4. Formulas of the form ϕ° are satisfied in the continuous semantics at the time $\tau(i)$ of an index i if they are satisfied in the pointwise semantics at i.

Corollary 6.10. *If $\lambda \vDash_{pw} \phi^\circ$ then $\lambda \vDash_c \phi^\circ$.*

That is, formulas of the form ϕ° are satisfied in the continuous semantics if they are satisfied in the pointwise semantics.

The other fragment of SAFETY-MTL that is of interest consists of the formulas that in the pointwise semantics are satisfied by all timed traces that satisfy them under the continuous semantics.

Definition 6.11 (C2PW). *Consider the sub-logic C2PW with the following syntax:*

$$\phi^\bullet ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \phi^\bullet \wedge \phi^\bullet \mid \phi^\bullet \vee \phi^\bullet \mid \phi^{\star\bullet}\ \mathcal{R}_{I^\forall}\ \phi^\bullet \mid \phi^\bullet\ \mathcal{U}_I\ \phi^{\star\bullet}$$

$$\phi^\star ::= \mathit{false} \mid a \mid \phi^\star \wedge \phi \mid \phi \wedge \phi^\star \mid \phi^\star \vee \phi^\star \mid \phi\ \mathcal{R}_{I^<}\ \phi^\star$$

$$\phi^{\star\bullet} ::= \mathit{false} \mid a \mid \phi^\bullet \wedge \phi^{\star\bullet} \mid \phi^{\star\bullet} \wedge \phi^\bullet \mid \phi^{\star\bullet} \vee \phi^{\star\bullet} \mid \phi^{\star\bullet}\ \mathcal{R}_{I^<}\ \phi^{\star\bullet}$$

where ϕ is any formula of SAFETY-*MTL, $I^<$ is of the form $[0,t)$ or $[0,t]$ or $[0,\infty)$, $I^\forall$ is any interval, and I is any bounded interval.*

Lemma 6.12. *Let $\lambda = \langle \sigma, \tau \rangle$ be a timed trace, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{\geq 0}$.*

1. *If $\lambda, t \vDash_c \phi^\star$ then there exists i such that $\tau(i) = t$.*
2. *If $\lambda, t \vDash_c \phi^{\star\bullet}$ then there exists i such that $\tau(i) = t$ and $\lambda, i \vDash_{pw} \phi^{\star\bullet}$.*
3. *If $\lambda, \tau(i) \vDash_c \phi^\bullet$ then $\lambda, i \vDash_{pw} \phi^\bullet$.*

That is:

1. Formulas of the form $\phi^\star$ are satisfied in the continuous semantics only at the instants of time that occur in the pointwise semantics.
2. Formulas of the form $\phi^{\star\bullet}$ are satisfied in the continuous semantics only at those instants of time in which they are satisfied in the pointwise semantics.
3. Formulas of the form $\phi^\bullet$ are satisfied in the pointwise semantics at an index i if they are satisfied in the continuous semantics at time $\tau(i)$.

Corollary 6.13. *If $\lambda \vDash_c \phi^\bullet$ then $\lambda \vDash_{pw} \phi^\bullet$.*

That is, formulas of the form $\phi^\bullet$ are satisfied in the pointwise semantics if they are satisfied in the continuous semantics (i.e., $\bullet$ is the dual of $\circ$).

Therefore, formulas that are both of the form ϕ° (in the language of PW2C) and $\phi^\bullet$ (in the language of C2PW) have the same continuous and pointwise semantics. We refer to the language of these formulas as SAFETY-MTL(R), the syntax of which can be found in the Appendix.

Theorem 6.14. *Given a set Φ of* SAFETY-*MTL(R) sentences and a sentence ϕ also of* SAFETY-*MTL(R):*

- *$\lambda \vDash_c \phi$ iff $\lambda \vDash_{pw} \phi$*
- *$\Lambda_\Phi^{pw} = \Lambda_\Phi^c$.*

As a consequence, for any set Φ of SAFETY-*MTL(R) sentences and sentence ϕ in* SAFETY-*MTL(R), we use $\lambda \vDash \phi$ instead of $\lambda \vDash_c \phi$ (or, equivalently, $\lambda \vDash_{pw} \phi$) and Λ_Φ instead of Λ_Φ^{pw} (or, equivalently, Λ_Φ^c).*
We additionally have that:

- *Λ_Φ is both r-closed and t-closed.*
- *If $\Lambda \vDash \Phi$ then $\Lambda^r \vDash \Phi$.*

- *If* $\Lambda \vDash \Phi$ *then* $\Lambda^t \vDash \Phi$.

That is, SAFETY-MTL(R) answers both Question 1 and Question 2.

As an example, consider the following MTL formula over the alphabet of process *CreditMgr* introduced in Sec. 3:

$$\phi = \mathit{processCredit}_{¡}\mathcal{R}(\neg \mathit{processCredit}_{¡} \vee \Diamond_{<4} \mathit{expectedDate}!)$$

where $\Diamond_{<t}\, \phi$ abbreviates $(\mathit{true}\, \mathcal{U}_{[0,t)}\, \phi)$. This formula specifies that *expectedDate* is published within four time units from the first delivery of *processCredit*. We show that this formula is both in the language of PW2C and C2PW.

1. ϕ is of the form ϕ°:
   (a) ϕ° has clause $\phi^{\circ}\ R_{I^{\forall}}\ \phi^{\diamond}$
   (b) $\mathit{processCredit}_{¡}$ is of the form ϕ° (third clause)
   (c) $(\neg \mathit{processCredit}_{¡} \vee (\mathit{true}\ \mathcal{U}_{[0,4)}\ \mathit{expectedDate}!))$ is of the form $\phi^{\diamond}$:
      i. $\phi^{\diamond}$ has clause $\phi^{\diamond} \vee \phi^{\circ}$
      ii. $\neg \mathit{processCredit}_{¡}$ is of the form $\phi^{\diamond}$ (third clause)
      iii. $\mathit{true}\ \mathcal{U}_{[0,4)}\ \mathit{expectedDate}!$ is of the form ϕ°:
         A. ϕ° has clause $\phi^{-}\ \mathcal{U}_I\ \phi^{\circ}$
         B. *true* is of the form ϕ^{-} (first clause)
         C. *expectedDate*! is of the form ϕ° (third clause)
         D. $[0, 4)$ is a bounded interval
2. ϕ is of the form $\phi^{\bullet}$:
   (a) $\phi^{\bullet}$ has clause $\phi^{\star\bullet}\ R_{I^{\forall}}\ \phi^{\bullet}$
   (b) $\mathit{processCredit}_{¡}$ is of the form $\phi^{\star\bullet}$ (second clause)
   (c) $(\neg \mathit{processCredit}_{¡} \vee (\mathit{true}\ \mathcal{U}_{[0,4)}\ \mathit{expectedDate}!))$ is of the form $\phi^{\bullet}$:
      i. $\phi^{\bullet}$ has clause $\phi^{\bullet} \vee \phi^{\bullet}$
      ii. $\neg \mathit{processCredit}_{¡}$ is of the form $\phi^{\bullet}$ (fourth clause)
      iii. $\mathit{true}\ \mathcal{U}_{[0,4)}\ \mathit{expectedDate}!$ is of the form $\phi^{\bullet}$:
         A. $\phi^{\bullet}$ has clause $\phi^{\bullet}\ \mathcal{U}_I\ \phi^{\star\bullet}$
         B. *true* is of the form $\phi^{\bullet}$ (first clause)
         C. *expectedDate*! is of the form $\phi^{\star\bullet}$ (second clause)
         D. $[0, 4)$ is a bounded interval

In a similar way, we could prove that the formula

$$\mathit{getClientRisk}_{¡}\mathcal{R}(\neg \mathit{getClientRisk}_{¡} \vee \Diamond_{<7}(\mathit{clientRiskValue}! \vee \mathit{clientRiskUnknown}!))$$

is also in the language of SAFETY-MTL(R). This formula specifies the behaviour of process *ClientsDB* we informally described in Sec. 3: when the first *getClientRisk* is delivered, it takes no more than seven time units to publish either *clientRiskValue* or *clientRiskUnknown*.

SAFETY-MTL(R) also allows us to specify the properties of the orchestrator *Clerk* we have formulated before:

- $creditReq ¡ \mathcal{R}(\neg creditReq ¡ \vee \Diamond_{<5} getClientRisk!)$ — *getClientRisk* is published within five time units from the first delivery of *creditReq*;
- $clientRiskValue ¡ \mathcal{R}(\neg clientRiskValue ¡ \vee \Diamond_{<10}(approved! \vee denied!))$ — either *approved* or *denied* are published within ten time units from the first delivery of *clientRiskValue*;
- $false \mathcal{R}(\neg approved! \vee (accept ¡ \mathcal{R}_{<5}(\neg accept ¡ \vee \Diamond_{<3} processCredit!)))$ — if *accept* is delivered within five time units of the publication of *approved*, *processCredit* is published within three time units;
- $expectedDate ¡ \mathcal{R}(\neg expectedDate ¡ \vee \Diamond_{<1} transferDate!)$ — *transferDate* is published within one time unit from the first delivery of *expectedDate*.

Some other examples of formulas in the language of SAFETY-MTL(R) are presented below. We use $\Diamond_I \phi$ and $\Box_I \phi$ to abbreviate $(true \, \mathcal{U}_I \, \phi)$ and $(false \, \mathcal{R}_I \, \phi)$, respectively.

- $\Box(\neg a \vee \Diamond_{\{7\}} b)$ — every execution of a is followed by an execution of b in exactly seven time units (e.g., setting of a timer and its timeout);
- $\Box(\neg a \vee (\Diamond_{(0,10)} c \vee \Diamond_{\{10\}} b))$ — every execution of a is followed by an execution of b in exactly ten time units unless c is executed first;
- $\Box(\neg a \vee (\Box_{(0,4)} \neg a \wedge \Diamond_{\{4\}} a))$ — after the first execution of a, it is executed regularly with a period of 4 time units.

6.3. Specifications

In this section, we work in SAFETY-MTL(R).

Definition 6.15 (Process specification)**.** *A specification of a process* $\langle \delta, \gamma, \Lambda \rangle$ *is* $\langle \mathsf{A}_\gamma, \Phi \rangle$ *such that* Φ *is in* SAFETY-*MTL(R) and* $\Lambda \subseteq \Lambda_\Phi$*, i.e., for every* $\lambda \in \Lambda$*,* $\lambda \vDash \Phi$*.*

As an example, consider the process *CreditMgr* informally described in Sec. 3 and assume that its behaviour Λ_m is the r-closure of the set of timed traces $\langle \sigma, \tau_{0.3} \rangle$ satisfying

$$\forall_{i \in \mathbb{N}} \; (processCredit ¡ \in \sigma(i) \wedge \forall_{j<i} \; processCredit ¡ \notin \sigma(j)) \Rightarrow \exists_{k>i} \; (expectedDate! \in \sigma(k) \wedge \tau(k) - \tau(i) < 4))$$

It is not difficult to prove that, for every $\lambda \in \Lambda_m$, $\lambda \vDash \phi$ where ϕ is the formula $(processCredit ¡ \mathcal{R}(\neg processCredit ¡ \vee \Diamond_{<4} expectedDate!)$ discussed above.

Given now a clock granularity δ and a specification $\langle \mathsf{A}_\gamma, \Phi \rangle$, we are interested in knowing if there is actually a process $\langle \delta, \gamma, \Lambda \rangle$ that it specifies. As already argued, the existence of such a process does not reduce to the consistency of the specification (i.e., the existence of a timed trace that satisfies Φ). This is because, although the set Λ_Φ is guaranteed to be r-closed, it is not necessarily the r-closure of a set of δ-timed traces, which is a requirement for Λ_Φ to be the behaviour of a process.

As also discussed at the beginning of this section, we consider instead the set

$$\Lambda_\Phi^\delta = \{\lambda : \lambda \in \mathsf{ttra}_\delta(\mathsf{A}_\gamma) \wedge \lambda \vDash \Phi\}^{tr}$$

It turns out that, in fact, $\{\lambda : \lambda \in \mathsf{ttra}_\delta(\mathsf{A}_\gamma) \wedge \lambda \vDash \Phi\}$ is already t-closed. This is because $\Lambda_\Phi = \{\lambda : \lambda \vDash \Phi\}$ is t-closed (Theo. 6.14) and $\mathsf{ttra}_\delta(\mathsf{A}_\gamma)$ is also t-closed (the intersection of t-closed sets being t-closed). Therefore,

$$\Lambda_\Phi^\delta = \{\lambda : \lambda \in \mathsf{ttra}_\delta(\mathsf{A}_\gamma) \wedge \lambda \vDash \Phi\}^r$$

Because the r-closure of the intersection of two sets is the intersection of their r-closures, and that Λ_Φ is r-closed (Theo. 6.14), we obtain

$$\Lambda_\Phi^\delta = \mathsf{ttra}_\delta(\mathsf{A}_\gamma)^r \cap \Lambda_\Phi$$

We also have that $\mathsf{ttra}_\delta(\mathsf{A}_\gamma)^r = \{\lambda = \langle \sigma, \tau\rangle : \tau \preceq \tau_\delta\}$, i.e., that set consists of all timed traces that refine a δ-timed trace. Therefore,

$$\Lambda_\Phi^\delta = \{\lambda = \langle \sigma, \tau\rangle : \tau \preceq \tau_\delta \wedge \lambda \vDash \Phi\}$$

is t-closed and the r-closure of a δ-timed property and, if it is not empty, $\langle \delta, \gamma, \Lambda_\Phi^\delta\rangle$ is indeed a process — the canonical process specified by $\langle \mathsf{A}_\gamma, \Phi\rangle$. To determine if Λ_Φ^δ is not empty, as discussed at the beginning of this section, one could build a DTIOM (which by definition generates only δ-timed traces) that satisfies Φ.

An alternative, which we explore here, is to identify a formula $\mathbb{A}x_\delta$ that is only satisfied by timed traces that are refinements of a δ-timed trace. In this case, one would check for the satisfiability of $\Phi \cup \{\mathbb{A}x_\delta\}$. Unfortunately, SAFETY-MTL(R) is not expressive enough to define such a formula (nor is MTL). Therefore, we introduce a new class of unary operators $\boxed{\delta}$ where $\delta \in \mathbb{Q}_{>0}$, which allow us to express that a timed trace contains all instants that are multiple of δ and that a sentence holds at all such instants:

$$\langle \sigma, \tau\rangle \vDash \boxed{\delta}\,\phi \;\text{ iff }\; \text{for all } n \in \mathbb{N}, \text{ there exists } j \text{ s.t. } \tau(j) = n \cdot \delta \text{ and } \langle \sigma, \tau\rangle, j \vDash \phi$$

Notice that restricting δ to $\mathbb{Q}_{>0}$ is not a real limitation. On the one hand, a connected HT-ARN is such that all the clock granularities are commensurate, which means that we can convert them to rational numbers by dividing them by a common divisor. On the other hand, reasoning about HT-ARNs that are not connected is not relevant because disconnected components do not interfere with each other. Notice that, for r-closure, one simply needs a dense set of time granularities.

In the extended language, the sentence $\boxed{\delta}(\Box_{(0,\delta)} \wedge_{a \in \mathsf{A}} \neg a)$ — where $\Box_{(0,t)}\ \phi$ is an abbreviation of ($\mathit{false}\ \mathcal{R}_{(0,t)}\ \phi$) — expresses a key property of δ-timed traces over A: empty observations occur at all time instants that are not multiple of δ. We denote this sentence by $\mathbb{A}x_\delta$ and, more generally, given $B \subseteq \mathsf{A}$, we use $\mathbb{A}x_\delta^B$ to denote $\boxed{\delta}(\Box_{(0,\delta)} \wedge_{a \in B} \neg a)$.

Proposition 6.16. *Let λ be a timed trace over A, and $\delta \in \mathbb{Q}_{>0}$: $\lambda \vDash \mathbb{A}x_\delta$ iff λ refines a δ-timed trace. That is, $\Lambda_{\mathbb{A}x_\delta} = \mathsf{ttra}_\delta(\mathsf{A})^r$.*

Corollary 6.17. *Let λ and λ' be timed traces such that $\lambda' \preceq \lambda$. If $\lambda \vDash \mathbb{A}x_\delta$ then $\lambda' \vDash \mathbb{A}x_\delta$.*

Proposition 6.18. *The sentence $\mathbb{A}x_\delta$ defines a divergent safe property.*

The following corollary allows us to make use of $\mathbb{A}x_\delta$ when deriving properties of a process specification. We use $\Phi \vdash \phi$ to mean that ϕ is a consequence of Φ, i.e., there is no timed trace λ such that $\lambda \not\models \phi$ and, for every $\phi' \in \Phi$, $\lambda \models \phi'$.

Corollary 6.19. *Let $\langle \mathsf{A}_\gamma, \Phi \rangle$ be a specification of $\langle \delta, \gamma, \Lambda \rangle$. If $\Phi, \mathbb{A}x_\delta \vdash \phi$ then, for every $\lambda \in \Lambda$, $\lambda \models \phi$.*

We are now interested in reasoning about properties of HT-ARNs. That is, given a HT-ARN α and a sentence ϕ in the language of A_α, we are interested in determining whether $\Lambda_\alpha \models \phi$, i.e., $\lambda \models \phi$ for every $\lambda \in \Lambda_\alpha$.

Proposition 6.20. *For every sentence ϕ in the language of* SAFETY-*MTL(R), $\Lambda_\alpha \models \phi$ iff $\Lambda_{P_\alpha} \models \phi$.*

This result, which is an immediate consequence of Prop. 6.6, ensures that the sentences satisfied by a HT-ARN α are precisely those that are satisfied by its best process approximation P_α (c.f. Theo. 4.3). However, one needs to calculate P_α explicitly to use this result; a more useful result is one that supports compositional reasoning as below.

Theorem 6.21. *Let α be a* HT-*ARN and, for every node (resp. hyperedge) p, let Φ_p be a specification of the process (resp. orchestrator) at p. Let*

$$\Phi_\alpha = \bigcup_{p \in N \cup E} \iota_p(\Phi_p \cup \mathbb{A}x_{\delta_p}^{\mathsf{A}_{\gamma_p}})$$

where the functions ι_p translate the sentences in the specification of the process at p (resp. of the orchestrator at c) to the language A_α. We have that $\Lambda_\alpha \models \phi$ if $\Phi_\alpha \vdash \phi$.

That is, to prove that ϕ expresses a property of α, it is sufficient to derive ϕ from specifications of the processes and orchestrators of α enriched with the corresponding $\mathbb{A}x_\delta$ axioms.

7. Related Work

Several researchers have recently addressed discrete timed systems with heterogeneous clock granularities. However, they have not focused on the development of theories of composability for such systems that can support compositional reasoning about the properties of the composition operation, which is what we addressed in this paper, especially in the context of run-time composition. An exception is [25], which studies when the composition of heterogeneous tag machines [4] is sound and complete. However, the notion of composition considered therein is more relaxed than ours, allowing for the delay between events to be modified. As a consequence, that notion is not appropriate for addressing global properties of systems interconnected at run time as actually implemented (i.e., implementations might have to be changed for the composition to work). By adopting instead a trace-based model in which composition corresponds to intersection, the model that we propose herein has the advantage of abstracting from the specificities of the different classes of automata that can be chosen as models of implementations. Because un-timed networks were investigated in [13] over traces, adopting a similar model for timed ones

also allows us to better appreciate the differences between un-timed and timed domains.

Formal clock calculi have also been developed that address heterogeneity, for example [15] in which a synchronous data-flow language is proposed that supports the modelling of multi-periodic systems and the refinement of clock granularities in a way that is similar to what we propose in this paper. However, the main focus of such calculi has been on modelling and simulation, not so much on the challenges that heterogeneity raises on run-time interconnection of systems; therefore, they are too specific on aspects that do not directly impact on system properties such as consistency. In fact, to the best of our knowledge, ours is the first model that adopts networks as components of systems and, therefore, addresses (run-time) compositionality at the network level.

Similarly, in [8], the authors introduce a formal communication model of behaviour for the composition of heterogeneous real-time cyber-physical systems based on logical clock constraints. Although this model supports the combination of heterogeneous timed systems, the authors do not consider the class of discrete periodic systems, which is particularly relevant for the processor-enabled systems that are now proliferating in cyberspace. To cope with heterogenous time scales, several approaches to the specification of real-time systems, notably the Timebands Framework [7], have also adopted an explicit representation of time granularity. However, neither composition (at design or run time) nor compositionality have been investigated in such frameworks.

Several frameworks have also been proposed for component/service-based software systems that exhibit timed properties, although not in a heterogeneous-time context. Algebraic frameworks such as [12, 19, 23, 28] address global properties similar to consistency, such as compatibility — i.e., that the conversation protocols (modelled as timed automata) followed by the peers in a choreography do not lead to deadlocks or time conflicts that prevent them from completing (e.g., reaching final states). However, the focus in this context is on the modelling of the (timed) conversation protocols that characterise the global behaviour of a (fixed) number of peers that exchange services. What this paper investigated is, instead, conditions through which we can guarantee that networks of components can work together when interconnected at run time to form larger networks. This has implications on the properties that need to be required of networks in order to guarantee consistency. An example is the way time is managed: in choreography, this is done globally for the (fixed) set of peers; in our approach, this needs to be done locally at level of each process because composition is dynamic.

The logical specification of properties of timed systems has been addressed since the beginning of the 90's, resulting in many different logics. Among them is MTL[24], upon which we built SAFETY-MTL(R) for expressing properties of heterogeneous networks of processes. The expressiveness of different logics, namely of variants and fragments of MTL (over both discrete and continuous time) has also been extensively addressed (e.g., [18, 21, 30]). Moreover, these logics have been studied for the purpose of verification (satisfiability, model-checking, bounded model-checking)[5, 26, 27]. The tension between the un-intituitive meaning of MTL with pontwise semantics and the decidability properties of MTL with continuous semantics have led Reynolds to propose 1CMTL [29], a new metric temporal logic that has a continuous semantics and a decision procedure for satisfiability. In this logic, temporal constraints are

associated with the reading of an arbitrary but not infinitely precise single universal clock. This logic is considered suitable for hybrid systems because it allows for imprecision in metric constraints.

In this paper we have defined a subset of MTL formulas that have the same semantics under continuous and pointwise semantics. In order to support the description of systems with physical and digital components in an unifying framework, Furia and Rossi have identified sufficient conditions under which formulas in the TRIO metric temporal logic have a consistent truth value when moving from continuous to discrete-time semantics and vice-versa [17]. These conditions rely on a class of *sampling invariant* formulas whose intending meaning is preserved when samplings of continuous behaviours are considered. In [14] we have also explored this way of relating continuous and discrete time domains, but the continuous interpretation over timed traces proposed in [30] revealed to be more appropriate for achieving the envisaged results.

As recognised in [16], properties such as '*a predicate p holds at all time instants that are multiple of* δ' are of interest in timed systems namely to express the behaviour of a clock signal. These properties, which can be expressed by $\boxed{\delta}\, p$ in the extension of SAFETY-MTL(R) that we defined in this paper, are not expressible in MTL and we are not aware of any extension of MTL that specifically deals with this problem (except that second-order extensions are known to solve it).

8. Concluding remarks

In this paper, we have proposed a component algebra for heterogenous timed systems that can be interconnected at run time. This algebra addresses the new class of systems that are beginning to operate in cyberspace where they connect dynamically to other systems to achieve some goal. Systems such as these often have real-time requirements, i.e., their correctness depends not only on what outputs are returned to given inputs, but also on the time at which inputs are received and corresponding outputs are produced and communicated. When software applications, usually written in a high-level programming language and relying on particular time abstractions, are deployed on a given platform, their real-time behaviour is additionally restricted by the clock period of that platform. Applications interconnected at run time across different platforms will be likely to operate over different clock periods, resulting in a timed heterogeneous system.

The elements of the proposed algebra are called HT-ARNs: multigraphs of nodes, each with its own clock granularity, where processes execute, and hyperedges where interactions among sets of such processes are orchestrated. Every hyperedge also has its own clock granularity, which needs to be a divisor of the clock granularities of the nodes that it connects so that they can interact. We provided compositionality results for ensuring the consistency of interconnections when performed at run time across different clock granularities. Contrarily to techniques that operate at design time (e.g., [4]), our results do not require changes to be performed on the processes that execute in such systems so that they can be interconnected, which would defeat the purpose of supporting dynamic binding.

Our algebra is based on timed traces, which allows us to abstract from the specificities of the different classes of automata that can be chosen as models of

implementations and characterise at a higher level the topological properties of the languages generated by such automata that support our compositionality results — in a companion paper [11] we investigate an algebra of automata-based machines, which we intend to extend to networks of automata.

Our results are based on a new time-related refinement relation and a new time-related closure operator that does not reduce to the Cantor topology of trace-based domains. Another area of further work concerns the logics that support an interface algebra for HT-ARNs: the logic that we proposed in the paper meets the requirements imposed by the new refinement and time-closure operators to make it suitable as a semantic domain, but further work is required to endow it with effective proof techniques.

Acknowledgments

This work was partially supported by the Engineering and Physical Sciences Research Council UK (EPSRC) through the grant EP/K000683/1, by the Royal Society International Exchange grant IE130154, and by the AFOSR grant FA9550-14-1-0043 "Semantic Completions: Unifying the Wave and the Particle Views of Information".

We would like to thank the reviewers for many useful comments and suggestions.

References

[1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284, 1991.

[2] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[3] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[4] A. Benveniste, B. Caillaud, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Tag machines. In *EMSOFT*, pages 255–263. ACM, 2005.

[5] M. M. Bersani, M. Rossi, and P. S. Pietro. A tool for deciding the satisfiability of continuous-time metric temporal logic. In *Proceedings of the 2013 20th International Symposium on Temporal Representation and Reasoning*, TIME '13, pages 99–106, Washington, DC, USA, 2013. IEEE Computer Society.

[6] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[7] A. Burns and I. J. Hayes. A timeband framework for modelling real-time systems. *Real-Time Syst.*, 45(1-2):106–142, June 2010.

[8] Y. Chen, Y. Chen, and E. Madelaine. Timed-pNets: a communication behavioural semantic model for distributed systems. *Frontiers of Computer Science*, 9(1):87–110, 2015.

[9] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Timed I/O automata: a complete specification theory for real-time systems. In *HSCC*, pages 91–100. ACM, 2010.

[10] B. Delahaye, J. L. Fiadeiro, A. Legay, and A. Lopes. A timed component algebra for services. In D. Beyer and M. Boreale, editors, *FORTE*, volume 7892 of *LNCS*, pages 242–257. Springer, 2013.

[11] B. Delahaye, J. L. Fiadeiro, A. Legay, and A. Lopes. Heterogeneous timed machines. In *ICTAC*, volume 8687 of *LNCS*, pages 115–132. Springer, 2014.

[12] G. Díaz, J. J. Pardo, M.-E. Cambronero, V. Valero, and F. Cuartero. Verification of web services with timed automata. *Electr. Notes Theor. Comput. Sci.*, 157(2):19–34, 2006.

[13] J. L. Fiadeiro and A. Lopes. An interface theory for service-oriented design. *Theor. Comput. Sci.*, 503:1–30, 2013.

[14] J. L. Fiadeiro and A. Lopes. Heterogeneous and asynchronous networks of timed systems. In S. Gnesi and A. Rensink, editors, *FASE*, volume 8411 of *LNCS*, pages 79–93. Springer, 2014.

[15] J. Forget, F. Boniol, D. Lesens, and C. Pagetti. A multi-periodic synchronous data-flow language. In *HASE*, pages 251–260. IEEE Computer Society, 2008.

[16] C. A. Furia, M. Pradella, and M. Rossi. Comments on temporal logics for real-time system specification. *ACM Comput. Surv.*, 41(2), 2009.

[17] C. A. Furia and M. Rossi. Integrating discrete- and continuous-time metric temporal logics through sampling. In E. Asarin and P. Bouyer, editors, *FORMATS*, volume 4202 of *LNCS*, pages 215–229. Springer, 2006.

[18] C. A. Furia and M. Rossi. On the expressiveness of MTL variants over dense time. In J.-F. Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *LNCS*, pages 163–178. Springer, 2007.

[19] N. Guermouche and C. Godart. Timed model checking based approach for web services analysis. In *ICWS*, pages 213–221. IEEE, 2009.

[20] T. A. Henzinger. Sooner is safer than later. *Inf. Process. Lett.*, 43(3):135–141, 1992.

[21] P. Hunter, J. Ouaknine, and J. Worrell. Expressive completeness for metric temporal logic. In *LICS*, pages 349–357. IEEE Computer Society, 2013.

[22] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers, 2006.

[23] R. Kazhamiakin, P. K. Pandya, and M. Pistore. Representation, verification, and computation of timed properties in web. In *ICWS*, pages 497–504. IEEE Computer Society, 2006.

[24] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[25] T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay. Tag machines for modeling heterogeneous systems. In *ACSD*, pages 186–195. IEEE Computer Society, 2013.

[26] J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *TACAS*, volume 3920 of *LNCS*, pages 411–425. Springer, 2006.

[27] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *FORMATS*, volume 5215 of *LNCS*, pages 1–13, 2008.

[28] J. Ponge, B. Benatallah, F. Casati, and F. Toumani. Analysis and applications of timed service protocols. *ACM Trans. Softw. Eng. Methodol.*, 19(4), 2010.

[29] M. Reynolds. A new metric temporal logic for hybrid systems. In *Proceedings of the 2013 20th International Symposium on Temporal Representation and Reasoning*, TIME '13, pages 73–80, Washington, DC, USA, 2013. IEEE Computer Society.

[30] D. Souza and P. Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *Int. J. Softw. Tools Technol. Transf.*, 9(1):1–4, Feb. 2007.

Appendix

Proof of Thm. 4.3

If a HT-ARN α is connected and $\Lambda_\alpha \neq \emptyset$, there is a process $P_\alpha = \langle \delta_\alpha, \gamma_\alpha, \Lambda_{P_\alpha} \rangle$ such that $\Lambda_{P_\alpha} \precapprox \Lambda_\alpha$ and, for any other process $P = \langle \delta, \gamma_\alpha, \Lambda_P \rangle$ such that $\Lambda_P \precapprox \Lambda_\alpha$, $\Lambda_P \precapprox \Lambda_{P_\alpha}$.

Proof. Let $P = \langle \delta, \gamma_\alpha, \Lambda_P \rangle$ be such that $\Lambda_P \precapprox \Lambda_\alpha$. Because $\Lambda_P \neq \emptyset$, let $\lambda \in \Lambda_P$; because $\Lambda_P \preceq \Lambda_\alpha$, we know that there is $\lambda' \in \Lambda_\alpha$ such that $\lambda \preceq_\rho \lambda'$, for some ρ. Because λ' necessarily contains all multiples of the clock granularities of the processes of α, so does λ, which implies that δ is a common divisor of all those clock granularities. Therefore, it makes sense to choose the granularity δ_α to be the greatest common divisor of the clock granularities of the processes of α.

The property Λ_{P_α} is constructed as follows:

- Let $\lambda = \langle \sigma, \tau \rangle \in \Lambda_\alpha$ be a trace of α.
  - Let p be a node or hyperedge of α.
    - We can project λ over the language of the process at p – $\lambda|_{\iota_p}$ – which we know to be a trace of Λ_p. By definition, there is a δ_p-timed trace λ_p in Λ_p such that $\lambda|_{\iota_p} \preceq \lambda_p$.
    - Because δ_p is a multiple of δ_α, we know by Prop. 2.5 that there is a single δ_α-timed trace $\lambda_p^* = \langle \sigma_p^*, \tau_{\delta_\alpha} \rangle$ that refines λ_p, which therefore belongs to Λ_p.
  - Let p be a node and $\lambda_p^\alpha = \langle \sigma_p^\alpha, \tau_{\delta_\alpha} \rangle$ be the timed trace with σ_p^α obtained from σ_p^* by prefixing all the actions with p. — this action sequence is in the language of A_α. We now build the trace $\lambda^\alpha = \langle \sigma^\alpha, \tau_{\delta_\alpha} \rangle$ such that $\sigma^\alpha(i) = \bigcup_{p \in N} \sigma_p^\alpha(i)$.
  - Because for all $p \in N$ the renaming ensures that unions are disjoint, $\lambda^\alpha|_{\iota_p} = \lambda_p^*$.
  - Taking now an hyperedge c, we know that, because σ is a trace over A_α, $\sigma_c(i) = \bigcup_{p \in c} \sigma_p(i)$. Because the refinement to δ_α adds $\emptyset$ to the new time instants, we conclude that $\lambda^\alpha|_{\iota_p} = \lambda_p^*$ also holds for all $p \in c$.
  - It then follows that $\lambda^\alpha|_{\iota_p} \in \Lambda_p$ for all $p \in N \cup E$, which by the definition of Λ_α implies that $\lambda^\alpha \in \Lambda_\alpha$.
- Let $\Lambda_{P_\alpha} = \{\lambda^\alpha : \lambda \in \Lambda_\alpha\}^r$.
  - It follows from the previous line that $\Lambda_{P_\alpha} \subseteq \Lambda_\alpha$ and, therefore, $\Lambda_{P_\alpha} \preceq \Lambda_\alpha$.
  - Given any $\lambda = \langle \sigma, \tau \rangle \in \Lambda_\alpha$, the refinements of λ and of λ^α to the meet of τ and τ_{δ_α} coincide. Therefore, $\Lambda_{P_\alpha} \precapprox \Lambda_\alpha$.

Notice that, because $\Lambda_\alpha \neq \emptyset$, $\Lambda_{P_\alpha} \neq \emptyset$; therefore, P_α is a process (and it approximates α.)

It remains to prove that P_α is the best approximation of α. For that purpose, let $P = \langle \delta, \gamma_\alpha, \Lambda_P \rangle$ be such that $\Lambda_P \precapprox \Lambda_\alpha$ We are going to prove that $\Lambda_P \precapprox \Lambda_{P_\alpha}$. Because we already know that δ_α is necessarily a multiple of δ, let n be δ_α/δ.

We start by remarking that, given $\lambda' = \langle \sigma', \tau' \rangle \in \Lambda_\alpha$ and $k \in \mathbb{N}$, if there is $i \in \mathbb{N}$ such $\tau'(i) = k.\delta_\alpha$, then $\sigma'^\alpha(k) = \sigma'(i)$, otherwise $\sigma'^\alpha(k) = \emptyset$.

- Let λ be a δ-timed trace of P.
  - Because $\Lambda_P \preceq \Lambda_\alpha$, we know that there is $\lambda' \in \Lambda_\alpha$ such that $\lambda \preceq_\rho \lambda'$ for some ρ.
  - It also follows that $\lambda \preceq \lambda'^\alpha$:
    - λ'^α is a δ_α-timed trace and δ_α is a multiple of δ, so $\tau \preceq \tau'^\alpha$;
    - for all i such that $i = k.n$ for some k, if i is in the range of ρ, then $\sigma(i) = \sigma'(\rho^{-1}(i)) = \sigma'^\alpha(k)$, otherwise $\sigma(i) = \sigma'^\alpha(k) = \emptyset$;
    - for any i that is not a multiple of n, $\sigma(i) = \emptyset$ because in this case $\tau(i)$ is not a multiple of the clocks of all processes in α.
  - Therefore, $\Lambda_P \preceq \Lambda_{P_\alpha}$.
- Consider now a trace $\lambda = \langle \sigma, \tau \rangle \in \Lambda_{P_\alpha}$.
  - We know that there is a trace $\lambda' \in \Lambda_\alpha$ such that $\lambda \preceq_\rho \lambda'^\alpha$ for some ρ. Since λ'^α also belongs to Λ_α,
  - Because $\Lambda_P \precapprox \Lambda_\alpha$, we know that there exists a trace $\lambda'' = \langle \sigma'', \tau'' \rangle \in \Lambda_P$ such that $\lambda'' \preceq_{\rho'} \lambda'$ for some ρ'.
  - Let λ''' be the refinement of λ'' to the meet of τ and τ'' (which implies that $\lambda''' \in \Lambda_P$).
  - We can prove that λ''' is also the refinement of λ to the meet of τ and τ'' as follows:

    (1) For all $j \in \mathbb{N}$ such that $\tau(j)$ is not a multiple of δ_α, since λ'^α is a δ_α-timed trace and $\lambda \preceq_\rho \lambda'^\alpha$, it follows that $\sigma(j) = \emptyset$. (a) If there is $k \in \mathbb{N}$ such that $\tau'(k) = \tau(j)$, since $\tau'(k)$ cannot be a multiple of the clocks of all processes of α, $\sigma'(k)$ must be $\emptyset$. In this case, $\tau''(\rho'(k)) = \tau'(k)$ and $\sigma''(\rho'(k)) = \emptyset$. Since λ''' is the refinement of λ'' to the meet of τ and τ'', for $m \in \mathbb{N}$ such that $\tau \wedge \tau''(m) = \tau''(\rho'(k))$, we have $\sigma'''(m) = \sigma''(\rho'(k))$. Hence we can conclude that, $\tau \wedge \tau''(m) = \tau(j)$ and $\sigma'''(m) = \emptyset$. (b) If for all $k \in \mathbb{N}$, $\tau'(k) \neq \tau(j)$ and there is l such that $\tau''(l) = \tau(j)$, then $\sigma''(l)$ is necessarily $\emptyset$. In this case, as before, we can conclude that for $m \in \mathbb{N}$ such that $\tau \wedge \tau''(m) = \tau''(l) = \tau(j)$, $\sigma'''(m) = \emptyset$. (c) If for all $l \in \mathbb{N}$, $\tau''(l) \neq \tau(j)$, since λ''' is the refinement of λ'' to the meet of τ and τ'', for $m \in \mathbb{N}$ such that $\tau \wedge \tau''(m) = \tau(j)$, $\sigma''(l)$ must be $\emptyset$.

    (2) For all $j \in \mathbb{N}$ such that $\tau(j)$ is a multiple of δ_α (and, hence, also multiple of δ), since λ'^α is a δ_α-timed trace, we have that $i = \rho^{-1}(j)$ is defined and $\sigma(j) = \sigma'^\alpha(i)$. (a) If there is $k \in \mathbb{N}$ such that $\tau'(k) = \tau(j)$ (and, hence, $\tau'(k) = \tau'^\alpha(i)$), as remarked above, $\sigma'^\alpha(i) = \sigma'(k)$. Since $\lambda'' \preceq_{\rho'} \lambda'$, $\tau''(\rho'(k)) = \tau'(k)$ and $\sigma''(\rho'(k)) = \sigma'(k)$. Since λ''' is the refinement of λ'' to the meet of τ and τ'', for $m \in \mathbb{N}$ such that $\tau \wedge \tau''(m) = \tau''(\rho'(k))$, we have $\sigma'''(m) = \sigma''(\rho'(k))$. Hence we can conclude that, $\tau \wedge \tau''(m) = \tau(j)$ and $\sigma'''(m) = \sigma(j)$. (b) If for all $k \in \mathbb{N}$, $\tau'(k) \neq \tau(j)$, $\tau(j)$ cannot be a multiple of the clocks of all processes of α and, hence, $\sigma(j) = \emptyset$. The result follows using the reasoning of 1(b) and 1(c) above.

Therefore, $\Lambda_P \precapprox \Lambda_{P_\alpha}$, i.e., P_α is the best approximation of α. □

Proof of Thm. 5.5

A HT-ARN is consistent if it is t-closed and progress-enabled.

Proof. Let $\alpha = \langle N, E, \Omega, \Xi \rangle$ be a t-closed HT-ARN and τ a time sequence relative to which it is progress-enabled. Given that Π_{α_τ} is not empty (it contains at least the empty segment ϵ), Π_{α_τ} can be organised as a (non-empty) tree. This tree is finitely branching because A_α is finite. Given that the HT-ARN is progress-enabled in relation to τ, the tree is infinite. By Kőnigs lemma, it contains an infinite branch σ.

We now prove that $\sigma \in \Lambda_{\alpha_\tau}$, i.e., $\sigma|_{\iota_p} \in \Lambda_{p_\tau}$ for all $p \in N \cup E$:

1. Let $p \in N \cup E$ and $\pi \prec \sigma|_{\iota_p}$. We know that π is of the form $\pi'|_{\iota_p}$ where $\pi' \in \Pi_{\alpha_\tau}$. Therefore, $\pi \in {\downarrow}\Lambda_{p_\tau}$.
2. It follows that $\sigma|_{\iota_p} \in \overline{\Lambda_{p_\tau}}$.
3. Because Λ_p is t-closed, we can conclude that $\sigma|_{\iota_p} \in \Lambda_{p_\tau}$.

□

Proof of Thm. 5.8

Let α be a composition of progress-enabled HT-ARNs through a family of wires with mutually-disjoint sets of interaction points, i.e.,

$$\alpha = (\alpha_1 \; \Big|\Big|^{i=1\ldots n}_{\langle\langle v_i, M_{v_i}\rangle, \xi_i, \langle p_i, M_{p_i}\rangle\rangle} \; \alpha_2)$$

where each $\langle\langle v_i, M_{v_i}\rangle, \xi_i, \langle p_i, M_{p_i}\rangle\rangle$ is a wire between α_1 and α_2. If the orchestrators involved in the wires (those that label the hyperedges c_{v_i}) are cooperative in relation to the free ports M_{v_i} that are being connected and the time granularity of the processes at the corresponding interaction points $\langle p_i, M_{p_i}\rangle$, and both HT-ARNs are delivery-enabled in relation to the interaction-points being connected, then α is progress-enabled.

Proof. To simplify the notation, we consider the case of a single wire with a connection interaction point from α_1, i.e., $\alpha = (\alpha_1 \,||_{\langle v, M_{c_v}\rangle, \xi, \langle p, M_p\rangle}\, \alpha_2)$ is a composition of progress-enabled HT-ARNs and $\alpha_{1_{c_v}}$ is a cooperative connection, α_1 is delivery-enabled in relation to $\langle v, M_{c_v}\rangle$ and α_2 is delivery-enabled in relation to $\langle p, M_p\rangle$. Also to simplify the notation, and since the two networks have disjoint sets of nodes, we rename the node v with p in α_1. Notice that, after this renaming, c_p denotes the hyperedge of α_1 that is changed by the composition, the connection interaction-point of α_1 becomes $\langle p, M_{c_p}\rangle$ and $\xi(\iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}})) = \iota_{c_p}(\mathsf{A}_{M_{c_p}}) = \iota_{2p}(\mathsf{A}_{M_p})$. We now prove that α is progress-enabled.

Let τ_1, τ_2 be time sequences relative to which α_1 and α_2 are progress-enabled, respectively. The idea of the proof is to show that α is progress-enabled in relation to any τ that refines $(\tau_1 \wedge \tau_2)$. Let τ be such a time sequence. Because

α_1 and α_2 are progress-enabled in relation to τ_1 and τ_2, respectively, they are also progress-enabled in relation to τ (which refines both).

We start by noticing that $\epsilon \in \Pi_{\alpha_\tau}$. This is because α_1 and α_2 are r-closed and progress-enabled in relation to τ_1 and τ_2, respectively.

Let now $\pi \in \Pi_{\alpha_\tau}$ and π_1, π_2 be the corresponding projections to the languages of α_1, α_2, respectively. We know that $\pi_1 \in \Pi_{\alpha_{1_\tau}}$ and $\pi_2 \in \Pi_{\alpha_{2_\tau}}$.

Because α_1 and α_2 are progress-enabled in relation to τ, let $(\pi_1 \cdot B_1) \in \Pi_{\alpha_{1_\tau}}$ and $(\pi_2 \cdot B_2) \in \Pi_{\alpha_{2_\tau}}$.

The change in the hyperedge c_p only interferes with the ability of the coordinator in c_p (from α_1) and the process in p (from α_2) to move — in α, the language of the coordinator only intersects that of the process p. Therefore, we need to adjust the deliveries in $B_2 \cap D_{\langle p, M_p \rangle}$ (the deliveries to the process in p through M_p) and the publications in $B_1 \cap D_{\langle p, M_{c_p} \rangle}$ (the publications by p that correspond to deliveries to the coordinator in c_p).

The proof proceeds by analysing different cases depending on whether the process in p and the coordinator in c_p synchronise. Let $k = \tau(|\pi|)$ be the time instant that follows the end of π — that at which B_1 and B_2 are executed. Since the attachment $\xi : M_{c_p} \to M_p$ of $\alpha_{1_{c_p}}$ to α_{2p} is well-formed we know that δ_p is a multiple of δ_{c_p}.

1. k is not a multiple of δ_{c_p}: In this case, we know that $B_1 \cap \iota_{1_{c_p}}(\mathsf{A}_{\gamma_{c_p}}) = \emptyset$. We also know that $B_2 \cap \iota_{2p}(\mathsf{A}_{\gamma_p}) = \emptyset$ because k is also not a multiple of δ_p. We have that $\pi \cdot B_1 \cup B_2 \in \Pi_{\alpha_\tau}$, because the projection over α_i is $(B_1 \cup B_2)|_{\iota_i} = B_1|_{\iota_i} \cup B_2|_{\iota_i} = B_i$ (neither B_1 intersects the language of α_2 nor B_2 intersects the language of α_1).

2. k is a multiple of δ_{c_p} but not of δ_p: In this case, we know that $B_2 \cap \iota_{2p}(\mathsf{A}_{\gamma_p}) = \emptyset$. We now prove that $\pi \cdot B_1' \cup B_2 \in \Pi_{\alpha_\tau}$ with $B_1' = B_1 \setminus \iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}})$.

projection over α_1 — We have that $(\pi_1 \cdot B_1) \in \Pi_{\alpha_{1_\tau}}$. Because $\alpha_{1_{c_p}}$ is a cooperative connection and α_1 is delivery-enabled in relation to $\langle p, M_{c_p} \rangle$, $\pi_1 \cdot B_1'$ is also in $\Pi_{\alpha_{1_\tau}}$ — the coordinator that orchestrates the connection $\alpha_{1_{c_p}}$ is ready to accept any set of publications at the node p and cannot force deliveries to a process that it connects when it is not in sync with it, so we can remove all deliveries to p from B_1; being publication-enabled, we can replace the publications that it was expecting from the process with those that it wants to do, which in this case is none (the processes not being in sync). On the other hand, $(B_1' \cup B_2)|_{\iota_1} = B_1'|_{\iota_1}$ because $B_2|_{\iota_1} = \emptyset$ (recall that $B_2 \cap \iota_p(\mathsf{A}_{M_p}) \subseteq B_2 \cap \iota_{2p}(\mathsf{A}_{\gamma_p}) = \emptyset$).

projection over α_2 — We have that $(\pi \cdot B_1' \cup B_2)|_{\iota_2}$ is $\pi_2 \cdot B_2$ because B_1' does not intersect $\iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}})$ and, hence, $\xi(B_1')$ does not intersect $\iota_{2p}(\mathsf{A}_{M_p})$ (recall that $\xi(\iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}})) = \iota_{2p}(\mathsf{A}_{M_p})$).

3. k is a multiple of δ_p: We prove that $\pi \cdot B_1' \cup B_2' \in \Pi_{\alpha_\tau}$ with $B_1' = B_1 \setminus \iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}}) \cup \xi((B_1 \cap \iota_{1_{c_p}}(\mathsf{A}_{M_{c_p}})) \setminus D_{\langle p, M_{c_p} \rangle})$ and $B_2' = B_2 \setminus D_{\langle p, M_p \rangle}$. In this way, (1) we remove from B_1 the deliveries that the coordinator c_p expects from p and consider only those that p wants to do (in B_2), which is possible because α_1 is delivery-enabled in relation to $\langle p, M_{c_p} \rangle$ and (2) we remove from B_2 the deliveries that p expects from the coordinator c_p and consider only those that the coordinator wants to do to p (the translation by ξ of those in B_1), which is possible because α_2 is delivery-enabled in relation to $\langle p, M_p \rangle$.

We have that:
$B_1 \cap \xi^{-1}(D_{\langle p,M_p\rangle}) = (B_1 \cap \iota_{1c_p}(\mathsf{A}_{M_{c_p}})) \setminus D_{\langle p,M_{c_p}\rangle}$
$B_2 \cap \xi(D_{\langle p,M_{c_p}\rangle}) = (B_2 \cap \iota_{2p}(\mathsf{A}_{M_p})) \setminus D_{\langle p,M_p\rangle}$

projection over α_1 — $(B'_1 \cup B'_2)|_{\iota_1} = B_1 \setminus D_{\langle p,M_{c_p}\rangle} \cup \xi^{-1}(B_2 \cap \xi(D_{\langle p,M_{c_p}\rangle}))$.

Because $(\pi_1 \cdot B_1) \in \Pi_{1_\tau}$ and α_1 is delivery-enabled in relation to $\langle p, M_{c_p}\rangle$ and
$\xi^{-1}(B_2 \cap \xi(D_{\langle p,M_{c_p}\rangle}))$ is a subset of $D_{\langle p,M_{c_p}\rangle}$, we know that $\pi_1 \cdot (B'_1 \cup B'_2)|_{\iota_1} \in \Pi_{1_\tau}$.

projection over α_2 — $(B'_1 \cup B'_2)|_{\iota_2} = B_2 \setminus D_{\langle p,M_p\rangle} \cup \xi(B_1 \cap \xi^{-1}(D_{\langle p,M_p\rangle}))$.

Because $(\pi_2 \cdot B_2) \in \Pi_{2_\tau}$ and α_2 is delivery-enabled in relation to $\langle p, M_p\rangle$ and
$\xi(B_1 \cap \xi^{-1}(D_{\langle p,M_p\rangle}))$ is a subset of $D_{\langle p,M_p\rangle}$, we know that $\pi_2 \cdot (B'_1 \cup B'_2)|_{\iota_2} \in \Pi_{2_\tau}$.

□

Proof of Prop. 6.2

Given a set Φ of sentences in SAFETY-MTL, $\Lambda^{pw}_\Phi = \{\lambda : \lambda \vDash_{pw} \Phi\}$ is divergent safe.

Proof. We need to prove that every trace that fails to satisfy a SAFETY-MTL formula ϕ has a finite bad prefix π, i.e., none of π's extensions satisfies ϕ. This follows from the following result:

if $\lambda, i \nvDash_{pw} \phi$, there is $i \leq k$ such that, for every $\lambda' : \lambda_k < \lambda'$, $\lambda', i \nvDash_{pw} \phi$

which can easily be proved by induction in the structure of SAFETY-MTL formulas. □

Proof of Prop. 6.3

Let Λ be a timed property. If Λ is divergent safe then it is also t-closed.

Proof. The fact that Λ is divergent safe means that for any timed trace λ, if for all $\pi < \lambda$ there is $\lambda' \in \Lambda$ such that $\pi < \lambda'$, then $\lambda \in \Lambda$. We need to prove that for every $\tau \in \Lambda_{time}$, Λ_τ is closed, i.e., that for any action sequence σ, if for all action segments $\pi' < \sigma$ there is $\sigma' \in \Lambda_\tau$ such that $\pi' < \sigma'$, then $\sigma \in \Lambda_\tau$.

Let σ be an action sequence such that for all action segments $\pi' < \sigma$ there is $\sigma' \in \Lambda_\tau$ such that $\pi' < \sigma'$. Then $\langle \sigma, \tau\rangle$ is a timed trace such that for all $\pi < \langle \sigma, \tau\rangle$ there is $\langle \sigma', \tau\rangle \in \Lambda$ such that $\pi < \langle \sigma', \tau\rangle$. Since Λ is divergent safe, we have that $\langle \sigma, \tau\rangle \in \Lambda$ and, hence, $\sigma \in \Lambda_\tau$. □

Proof of Prop. 6.6

Given timed traces λ and λ' such that $\lambda' \preceq \lambda$, $\lambda \vDash_c \phi$ iff $\lambda' \vDash_c \phi$. It follows that, for every Φ, $\Lambda^c_\Phi = \{\lambda : \lambda \vDash_c \Phi\}$ is r-closed.

Proof. Let $\lambda = \langle \sigma, \tau \rangle$ and $\lambda' = \langle \sigma', \tau' \rangle$ be two timed traces. We prove by induction on the structure of the formula that, if $\lambda' \preceq_\rho \lambda$, for some ρ then, for every $t \in \mathbb{R}_{\geq 0}$, $\lambda, t \vDash_c \phi$ iff $\lambda', t \vDash_c \phi$:

a : By definition, $\lambda, t \vDash_c \phi$ iff there exists $i \in \mathbb{N}$ such that $\tau(i) = t$ and $a \in \sigma(i)$ and $\lambda', t \vDash_c \phi$ iff there exists $j \in \mathbb{N}$ such that $\tau'(j) = t$ and $a \in \sigma'(j)$.

$\Rightarrow$ If there exists $i \in \mathbb{N}$ such that $\tau(i) = t$ and $a \in \sigma(i)$, we take $j = \rho(i)$ since $\sigma'(\rho(i)) = \sigma(i)$.

$\Leftarrow$ If there exists $j \in \mathbb{N}$ such that $\tau'(j) = t$ and $a \in \sigma'(j)$, then $\sigma'(j)$ is not empty. Because $\lambda' \preceq \lambda$, then there exists $i \in \mathbb{N}$ such that $\rho(i) = j$ and, hence, the result follows immediately.

$\neg\phi$: It follows straightforwardly from the induction hypothesis.

$\phi_1 \supset \phi_2$: It follows straightforwardly from the induction hypothesis.

$\phi_1 \; \mathcal{U}_I \; \phi_2$: It follows straightforwardly from the induction hypothesis.

□

An observation useful in the proof of Lemma 6.9 is the following:

Lemma 8.1. *Given a time sequence τ and $t \in \mathbb{R}_{\geq 0}$, the following conditions are equivalent:*

- *there exists $i \in \mathbb{N}$ such that $t = \tau(i)$;*
- *$next_\tau(t) = t$;*
- *$previous_\tau(t) = t$.*

Proof. This follows straightforwardly from Def. 6.7. □

Proof of Lemma 6.9

Let $\lambda = \langle \sigma, \tau \rangle$ be a timed trace, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{\geq 0}$.

1. If for all $i \in \mathbb{N}$ $\tau(i) \neq t$ then $\lambda, t \vDash_c \phi^\diamond$
2. If $\lambda, next_\lambda(t) \vDash_{pw} \phi^+$ then $\lambda, t \vDash_c \phi^+$.
3. If $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ then $\lambda, t \vDash_c \phi^-$.
4. If $\lambda, i \vDash_{pw} \phi^\circ$ then $\lambda, \tau(i) \vDash_c \phi^\circ$.

Proof. We prove (1)-(4) by simultaneous induction on the number of connectives that occur in the formulas $\phi^\diamond, \phi^+, \phi^-$ and ϕ°.

Base case:

$\phi^\diamond$ **is** *true***:** Follows trivially from the fact that, for every t, $\lambda, t \vDash_c \mathit{true}$.

ϕ^+ **is** *true***:** Similar to the previous case.

ϕ^+ **is** *false***:** By Def. 6.1, $\lambda, \mathit{next}_\lambda(t) \nvDash_{pw} \mathit{false}$.

ϕ^- **is** *true***:** Similar to the first case.

ϕ^- **is** *false***:** By Def. 6.1, $\lambda, \mathit{previous}_\lambda(t) \nvDash_{pw} \mathit{false}$.

ϕ° **is** *true***:** Similar to the first case.

ϕ° **is** *false***:** By Def. 6.1, $\lambda, i \nvDash_{pw} \mathit{false}$.

ϕ° **is** a**:** By Def. 6.1, $\lambda, i \vDash_{pw} a$ implies that $a \in \sigma(i)$ and, hence, by Def. 6.5, $\lambda, \tau(i) \vDash_c a$.

Inductive step:

$\phi^\diamond$ **is** $\neg a$**:** By Def. 6.5, the fact that, for all $i \in \mathbb{N}$, $\tau(i) \neq t$ implies $\lambda, t \vDash_c \neg a$.

$\phi^\diamond$ **is** $\phi_1^\diamond \wedge \phi_2^\diamond$**:** If, for all $i \in \mathbb{N}$, $\tau(i) \neq t$, then by the induction hypothesis of (1) we have that $\lambda, t \vDash_c \phi_1^\diamond$ and $\lambda, t \vDash_c \phi_2^\diamond$ and, hence, by Def. 6.5, $\lambda, t \vDash_c \phi^\diamond$.

$\phi^\diamond$ **is** $\phi_1^\diamond \vee \phi_2^\diamond$**:** If for all $i \in \mathbb{N}$, $\tau(i) \neq t$, then by the induction hypothesis of (1) we have that $\lambda, t \vDash_c \phi_1^\diamond$ and, hence, by Def. 6.5, $\lambda, t \vDash_c \phi^\diamond$.

$\phi^\diamond$ **is** $\phi_1^\circ \vee \phi_2^\diamond$**:** Similar to the previous case.

ϕ^+ **is** $\neg a$**:** By Def. 6.1, $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \neg a$ implies that $a \notin \sigma(\mathit{next}_\lambda(t))$. If $\tau(\mathit{next}_\lambda(t)) = t$ then, by Def. 6.5, $\lambda, t \vDash_c \neg a$. If $\tau(\mathit{next}_\lambda(t)) \neq t$, then, for all $i \in \mathbb{N}$, $\tau(i) \neq t$ and it follows that $\lambda, t \vDash_c \neg a$.

ϕ^+ **is** $\phi_1^+ \wedge \phi_2^+$**:** By Def. 6.1, $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi^+$ implies that $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi_1^+$ and $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi_2^+$. By the induction hypothesis of (2), we have that $\lambda, t \vDash_c \phi_1^+$ and $\lambda, t \vDash_c \phi_2^+$ and, hence, $\lambda, t \vDash_c \phi^+$.

ϕ^+ **is** $\phi_1^+ \vee \phi_2^+$**:** By Def. 6.1, $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi^+$ implies that $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi_1^+$ or $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi_2^+$. By the induction hypothesis of (2), we have that $\lambda, t \vDash_c \phi_1^+$ or $\lambda, t \vDash_c \phi_2^+$ and, hence, $\lambda, t \vDash_c \phi^+$.

ϕ^+ **is** $\phi_1^\circ \mathcal{R} \phi_2^+$**:** By Def. 6.1, $\lambda, \mathit{next}_\lambda(t) \vDash_{pw} \phi^+$ implies that, for all $k \geq \mathit{next}_\lambda(t)$, either $\lambda, k \vDash_{pw} \phi_2^+$ or there exists j such that $\mathit{next}_\lambda(t) \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. Let $t' \geq t$ and $k = \mathit{next}_\lambda(t')$. Since $k \geq \mathit{next}_\lambda(t)$, we have that either $\lambda, \mathit{next}_\lambda(t') \vDash_{pw} \phi_2^+$ or there exists j such that $\mathit{next}_\lambda(t) \leq j < \mathit{next}_\lambda(t')$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. In the first case, by the induction hypothesis of (2), we have that $\lambda, t' \vDash_c \phi_2^+$. In the second case, by the induction hypothesis of (4), we have that $\lambda, \tau(j) \vDash_c \phi_1^\circ$. Let $r = \tau(j)$. Since $\mathit{next}_\lambda(t) \leq j < \mathit{next}_\lambda(t')$ implies $t \leq \tau(j) < t'$, then we have $t \leq r < t'$ and $\lambda, r \vDash_c \phi_1^\circ$. We can then conclude that $\lambda, t \vDash_c \phi^+$.

ϕ^- **is** $\neg a$**:** By Def. 6.1, if $\lambda, \mathit{previous}_\lambda(t) \vDash_{pw} \neg a$, $a \notin \sigma(\mathit{previous}_\lambda(t))$. If $\tau(\mathit{previous}_\lambda(t)) = t$, then $\lambda, t \vDash_c \neg a$. If $\tau(\mathit{next}_\lambda(t)) \neq t$, then for all $i \in \mathbb{N}$, $\tau(i) \neq t$ and it follows that $\lambda, t \vDash_c \neg a$.

ϕ^- **is** $\phi_1^- \wedge \phi_2^-$**:** By Def. 6.1, we have that $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ implies that $\lambda, previous_\lambda(t) \vDash_{pw} \phi_1^-$ and $\lambda, previous_\lambda(t) \vDash_{pw} \phi_2^-$. By the induction hypothesis of (3), we have that $\lambda, t \vDash_c \phi_1^-$ and $\lambda, t \vDash_c \phi_2^-$ and, hence, $\lambda, t \vDash_c \phi^-$.

ϕ^- **is** $\phi_1^- \vee \phi_2^-$**:** By Def. 6.1, we have that $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ implies that $\lambda, previous_\lambda(t) \vDash_{pw} \phi_1^-$ or $\lambda, previous_\lambda(t) \vDash_{pw} \phi_2^-$. By the induction hypothesis of (3), we have that $\lambda, t \vDash_c \phi_1^-$ or $\lambda, t \vDash_c \phi_2^-$ and, hence, $\lambda, t \vDash_c \phi^-$.

ϕ^- **is** $\phi_1^\circ \mathcal{R} \phi_2^-$**:** By Def. 6.1, $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ implies that, for all $k \geq previous_\lambda(t)$, either $\lambda, k \vDash_{pw} \phi_2^-$ or there exists j such that $previous_\lambda(t) \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. Let $t' \geq t$ and $k = previous_\lambda(t')$. Since $k \geq previous_\lambda(t)$, either λ, $previous_\lambda(t') \vDash_{pw} \phi_2^-$ or there exists j such that $previous_\lambda(t) \leq j < previous_\lambda(t')$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. In the first case, by the induction hypothesis of (3), we have that $\lambda, t' \vDash_c \phi_2^-$. In the second case, by the induction hypothesis of (4), we have that $\lambda, \tau(j) \vDash_c \phi_1^\circ$. Let $r = \tau(j)$. Since $previous_\lambda(t) \leq j < previous_\lambda(t')$ implies $t \leq \tau(j) < t'$, then we have $t \leq r < t'$ and $\lambda, r \vDash_c \phi_1^\circ$. We can then conclude that $\lambda, t \vDash_c \phi^-$.

ϕ^- **is** $\phi_1^- \mathcal{U}_{I_0^-} \phi_2^\circ$**:** By Def. 6.1, $\lambda, previous_\lambda(t) \vDash_{pw} \phi^-$ implies that there exists $k \geq previous_\lambda(t)$ such that $(\tau(k) - \tau(previous_\lambda(t))) \in I_0^-$, $\lambda, k \vDash_{pw} \phi_2^\circ$ and, for all j such that $previous_\lambda(t) \leq j < k$, $\lambda, j \vDash_{pw} \phi_1^-$. Let $t' = \tau(k)$. On the one hand, we have that $\tau(previous_\lambda(t)) \leq t$ and, hence, $(t' - t) \leq (t' - \tau(previous_\lambda(t)))$. On the other hand, since $0 \notin I_0^-$, $k > previous_\lambda(t)$ and, hence, $t' > t$. The fact that I_0^- is of the form $(0, u)$ or $(0, u]$ allows us to conclude that $(t' - t) \in I_0^-$. By the induction hypothesis of (4), we additionally have that $\lambda, t' \vDash_c \phi_2^\circ$. Let r be such that $t \leq r < t'$. We have that $previous_\lambda(t) \leq previous_\lambda(r) < k$. Hence, we can apply the induction hypothesis of (3), which ensures that $\lambda, r \vDash_c \phi_1^-$.

ϕ° **is** $\neg a$**:** By Def. 6.1, if $\lambda, i \vDash_{pw} \neg a$, we have that $a \notin \sigma(i)$. Then, by Def. 6.5, $\lambda, \tau(i) \nvDash_c a$ and, hence, $\lambda, \tau(i) \vDash_c \neg a$.

ϕ° **is** $\phi_1^\circ \wedge \phi_2^\circ$**:** By Def. 6.1, we have that $\lambda, i \vDash_{pw} \phi^\circ$ implies that $\lambda, i \vDash_{pw} \phi_1^\circ$ and $\lambda, i \vDash_{pw} \phi_2^\circ$. By the induction hypothesis of (4), we have that $\lambda, \tau(i) \vDash_c \phi_1^\circ$ and $\lambda, \tau(i) \vDash_c \phi_2^\circ$ and, hence, $\lambda, \tau(i) \vDash_c \phi^\circ$.

ϕ° **is** $\phi_1^\circ \vee \phi_2^\circ$**:** By Def. 6.1, we have that $\lambda, i \vDash_{pw} \phi^\circ$ implies that $\lambda, i \vDash_{pw} \phi_1^\circ$ or $\lambda, i \vDash_{pw} \phi_2^\circ$. By the induction hypothesis of (4), we have that $\lambda, \tau(i) \vDash_c \phi_1^\circ$ or $\lambda, \tau(i) \vDash_c \phi_2^\circ$ and, hence, $\lambda, \tau(i) \vDash_c \phi^\circ$.

ϕ° **is** $\phi_1^\circ \mathcal{R}_{I^<} \phi_2^-$**:** By Def. 6.1, $\lambda, i \vDash_{pw} \phi^\circ$ implies that, for all $k \geq i$ such that $(\tau(k) - \tau(i)) \in I^<$, either $\lambda, k \vDash_{pw} \phi_2^-$ or there exists j such that $i \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. Let $t \geq \tau(i)$ such that $(t - \tau(i)) \in I^<$ and $k = previous_\lambda(t)$. It follows that $k \geq i$ and that $\tau(k) \leq t$, which implies $(\tau(k) - \tau(i)) \leq (t - \tau(i))$. Since $I^<$ is any interval starting in 0, $(\tau(k) - \tau(i)) \in I^<$. Then, we have that either $\lambda, previous_\lambda(t) \vDash_{pw} \phi_2^-$ or there exists j such that $i \leq j < previous_\lambda(t)$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. In the first case, by the induction hypothesis of (3), we have that $\lambda, t \vDash_c \phi_2^-$. In the second case, by the induction hypothesis of (4),

we have that $\lambda, \tau(j) \vDash_c \phi_1^\circ$. Let $r = \tau(j)$. Since $i \leq j < previous_\lambda(t)$ implies $\tau(i) \leq \tau(j) < t$, then we have $\tau(i) \leq r < t$ and $\lambda, r \vDash_c \phi_1^\circ$. We can then conclude that $\lambda, \tau(i) \vDash_c \phi^\circ$.

ϕ° **is** $\phi_1^\circ \mathcal{R}_{I^+} \phi_2^+$**:** By Def. 6.1, $\lambda, i \vDash_{pw} \phi^\circ$ implies that, for all $k \geq i$ such that $(\tau(k) - \tau(i)) \in I^+$, either $\lambda, k \vDash_{pw} \phi_2^+$ or there exists j such that $i \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. Let $t \geq \tau(i)$ such that $(t - \tau(i)) \in I^+$ and $k = next_\lambda(t)$. It follows that $k \geq i$ and $\tau(k) \geq t$, which implies $(\tau(k) - \tau(i)) \geq (t - \tau(i))$. Since I^+ is an unbounded interval, $(\tau(k) - \tau(i)) \in I^+$. Then, we have that either $\lambda, next_\lambda(t) \vDash_{pw} \phi_2^+$ or there exists j such that $i \leq j < next_\lambda(t)$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. In the first case, by the induction hypothesis of (2), we have that $\lambda, t \vDash_c \phi_2^+$. In the second case, by the induction hypothesis of (4), we have that $\lambda, \tau(j) \vDash_c \phi_1^\circ$. Let $r = \tau(j)$. Since $i \leq j < next_\lambda(t)$ implies $\tau(i) \leq \tau(j) < t$, then we have $\tau(i) \leq r < t$ and $\lambda, r \vDash_c \phi_1^\circ$. We can then conclude that $\lambda, \tau(i) \vDash_c \phi^\circ$.

ϕ° **is** $\phi_1^\circ \mathcal{R}_{I^\forall} \phi_2^\diamond$**:** By Def. 6.1, $\lambda, i \vDash_{pw} \phi^\circ$ implies that, for all $k \geq i$ such that $(\tau(k) - \tau(i)) \in I^\forall$, either $\lambda, k \vDash_{pw} \phi_2^\diamond$ or there exists j such that $i \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. Let $t \geq \tau(i)$ such that $(t - \tau(i)) \in I^\forall$. If, for all $i \in \mathbb{N}$, $\tau(i) \neq t$, then by the induction hypothesis of (1), $\lambda, t \vDash_c \phi_2^\diamond$. If there exists k such that $t = \tau(k)$, we have that either $\lambda, k \vDash_{pw} \phi_2^\diamond$ or there exists j such that $i \leq j < k$ and $\lambda, j \vDash_{pw} \phi_1^\circ$. It is easy to see that all $\diamond$-formulas are also $\circ$-formulas and, hence, in the first case, we can apply the induction hypothesis of (4) to conclude that $\lambda, t \vDash_c \phi_2^\diamond$. In the second case, also by the induction hypothesis of (4), we have that $\lambda, \tau(j) \vDash_c \phi_1^\circ$. Let $r = \tau(j)$. Since $i \leq j < k$ implies $\tau(i) \leq \tau(j) < t$, then we have $\tau(i) \leq r < t$ and $\lambda, r \vDash_c \phi_1^\circ$. We can then conclude that $\lambda, \tau(i) \vDash_c \phi^\circ$.

ϕ° **is** $\phi_1^- \mathcal{U}_I \phi_2^\circ$**:** By Def. 6.1, $\lambda, i \vDash_{pw} \phi^\circ$ implies that there exists $k \geq i$ such that $(\tau(k) - \tau(i)) \in I$, $\lambda, k \vDash_{pw} \phi_2^\circ$ and, for all j such that $i \leq j < k$, $\lambda, j \vDash_{pw} \phi_1^-$. Let $t' = \tau(k)$. We have that $t' \geq \tau(i)$ and, by the induction hypothesis of (4), $\lambda, t' \vDash_c \phi_2^\circ$. Let r be such that $\tau(i) \leq r < t' = \tau(k)$. We have that $i \leq previous_\lambda(r) < k$. Hence, we can apply the induction hypothesis of (3), which ensures that $\lambda, r \vDash_c \phi_1^-$. We can then conclude that $\lambda, \tau(i) \vDash_c \phi^\circ$.

□

Proof of Corollary 6.10

If $\lambda \vDash_{pw} \phi^\circ$ then $\lambda \vDash_c \phi^\circ$.

Proof. This follows from (4) of Lemma 6.9. By Def. 6.1, $\lambda \vDash_{pw} \phi^\circ$ iff $\lambda, 0 \vDash_{pw} \phi^\circ$. By definition of timed trace $\tau(0) = 0$ and, hence, $\lambda, 0 \vDash_c \phi^\circ$. By Def. 6.5, we then conclude $\lambda \vDash_c \phi^\circ$. □

Proof of Lemma 6.12

Let $\lambda = \langle \sigma, \tau \rangle$ be a timed trace, $i \in \mathbb{N}$ and $t \in \mathbb{R}_{\geq 0}$.

1. If $\lambda, t \models_c \phi^\star$ then there exists i such that $\tau(i) = t$.
2. If $\lambda, t \models_c \phi^{\star\bullet}$ then there exists i such that $\tau(i) = t$ and $\lambda, i \models_{pw} \phi^{\star\bullet}$.
3. If $\lambda, \tau(i) \models_c \phi^\bullet$ then $\lambda, i \models_{pw} \phi^\bullet$.

Proof. We prove (1)-(3) by simultaneous induction on the number of connectives that occur in the formulas $\phi^\star, \phi^{\star\bullet}$ and $\phi^\bullet$.

Base case:

$\phi^\star$ **is** *false***:** The result follows from the fact that, by Def. 6.5, $\lambda, t \not\models_c$ *false* for every t.

$\phi^\star$ **is** a**:** The result follows from the fact that, by Def. 6.5, $\lambda, t \models_c a$ implies that there exists i such tthat $t = \tau(i)$.

$\phi^{\star\bullet}$ **is** *false***:** Similar to the case of $\phi^\star$.

$\phi^{\star\bullet}$ **is** a**:** The result follows from the fact that, by Def. 6.1, $\lambda, t \models_c a$ implies that there exists i such that $\tau(i) = t$ and $a \in \sigma(i)$. Hence, by Def. 6.1, $\lambda, i \models_{pw} a$.

$\phi^\bullet$ **is** *false***:** Similar to the case of $\phi^\star$.

$\phi^\bullet$ **is** *true***:** The result follows from the fact that, by Def. 6.1, for every t, $\lambda, t \models_c$ *true*.

$\phi^\bullet$ **is** a**:** By Def. 6.5, $\lambda, \tau(i) \models_c a$ implies that $a \in \sigma(i)$ and, hence, by Def. 6.1, $\lambda, i \models_{pw} a$.

Inductive step:

$\phi^\star$ **is** $\phi_1^\star \wedge \phi_2$**:** By Def. 6.5, $\lambda, t \models_c \phi_1^\star$. By the induction hypothesis of (1), we have that there exists i such that $t = \tau(i)$.

$\phi^\star$ **is** $\phi_1 \wedge \phi_2^\star$**:** Similar to the previous case.

$\phi^\star$ **is** $\phi_1^\star \vee \phi_2^\star$**:** By Def. 6.5, either $\lambda, t \models_c \phi_1^\star$ or $\lambda, t \models_c \phi_2^\star$. By the induction hypothesis of (1), in both cases, there exists i such that $t = \tau(i)$.

$\phi^\star$ **is** $\phi_1 \mathcal{R}_{I^<} \phi_2^\star$**:** By Def. 6.5, $\lambda, t \models_c \phi^\star$ implies that, for all $u \geq t$ such that $(u - t) \in I^<$, either $\lambda, u \models_c \phi_2^\star$ or there exists r such that $t \leq r < u$ and $\lambda, r \models_c \phi_1$. Because $0 \in I^<$, this also holds for $u = t$, i.e., either $\lambda, t \models_c \phi_2^\star$ or there exists r such that $t \leq r < t$ and $\lambda, r \models_c \phi_1$. Because there is no r such that $t \leq r < t$, it follows that $\lambda, t \models_c \phi_2^\star$. By the induction hypothesis of (1), there exists i such that $t = \tau(i)$.

$\phi^{\star\bullet}$ **is** $\phi_1^\bullet \wedge \phi_2^{\star\bullet}$**:** By Def. 6.5, $\lambda, t \models_c \phi^{\star\bullet}$ implies that $\lambda, t \models_c \phi_1^\bullet$ and $\lambda, t \models_c \phi_2^{\star\bullet}$. By the induction hypothesis of (2), we have that there exists i such that $t = \tau(i)$ and $\lambda, i \models_{pw} \phi_2^{\star\bullet}$. Then, we are also in the conditions of applying the induction hypothesis of (3), and can conclude that $\lambda, i \models_{pw} \phi_1^\bullet$. Hence, there exists i such that $t = \tau(i)$ and $\lambda, i \models_c \phi^{\star\bullet}$.

$\phi^{\star\bullet}$ **is** $\phi_1^{\star\bullet} \wedge \phi_2^\bullet$**:** Similar to the previous case.

$\phi^{\star\bullet}$ **is** $\phi_1^{\star\bullet} \vee \phi_2^{\star\bullet}$**:** By Def. 6.5, either $\lambda, t \vDash_c \phi_1^{\star\bullet}$ or $\lambda, t \vDash_c \phi_2^{\star\bullet}$. By the induction hypothesis of (2), we have that there exists i such that $t = \tau(i)$ and either $\lambda, i \vDash_{pw} \phi_1^{\star\bullet}$ or $\lambda, i \vDash_{pw} \phi_2^{\star\bullet}$. Therefore, $\lambda, i \vDash_{pw} \phi^{\star\bullet}$.

$\phi^{\star\bullet}$ **is** $\phi_1^{\star\bullet} \mathcal{R}_{I^<} \phi_2^{\star\bullet}$**:** By Def. 6.5, $\lambda, t \vDash_c \phi^{\star\bullet}$ implies that, for all $u \geq t$ such that $(u - t) \in I^<$, either $\lambda, u \vDash_c \phi_2^{\star\bullet}$ or there exists r such that $t \leq r < u$ and $\lambda, r \vDash_c \phi_1^{\star\bullet}$ (†). In particular, since $0 \in I^<$, this is also true for $u = t$. Because it does not exist a r such that $t \leq r < t$, it follows that $\lambda, t \vDash_c \phi_2^{\star\bullet}$. By the induction hypothesis of (2), we have that there exists i such that $t = \tau(i)$. Let $k \geq i$ such that $(\tau(k) - \tau(i)) \in I^<$. We are in the conditions of applying (†) to $u = \tau(k)$ and, hence, either $\lambda, \tau(k) \vDash_c \phi_2^{\star\bullet}$ or there exists r such that $\tau(i) \leq r < \tau(k)$ and $\lambda, r \vDash_c \phi_1^{\star\bullet}$. In the first case, by the induction hypothesis of (2), we have that $\lambda, k \vDash_{pw} \phi_2^{\star\bullet}$. In the second case, by the induction hypothesis of (2) we have that there exists j such that $\tau(j) = r$ and $\lambda, j \vDash_{pw} \phi_1^{\star\bullet}$. To conclude that $\lambda, i \vDash_{pw} \phi^{\star\bullet}$, it remains to ensure that $i \leq j < k$, which follows from the fact that $\tau(i) \leq r < \tau(k)$.

$\phi^\bullet$ **is** $\neg a$**:** By Def. 6.5, $\lambda, \tau(i) \vDash_c \neg a$ implies that, $a \notin \sigma(i)$. By Def. 6.1, $\lambda, i \vDash_{pw} \neg a$.

$\phi^\bullet$ **is** $\phi_1^\bullet \wedge \phi_2^\bullet$**:** By Def. 6.5, $\lambda, \tau(i) \vDash_c \phi^\bullet$ implies that $\lambda, \tau(i) \vDash_c \phi_1^\bullet$ and $\lambda, \tau(i) \vDash_c \phi_2^\bullet$. By the induction hypothesis of (3), if follows that $\lambda, i \vDash_{pw} \phi_1^\bullet$ and $\lambda, i \vDash_{pw} \phi_2^\bullet$. Therefore, $\lambda, i \vDash_c \phi^\bullet$.

$\phi^\bullet$ **is** $\phi_1^\bullet \vee \phi_2^\bullet$**:** By Def. 6.5, either $\lambda, \tau(i) \vDash_c \phi_1^\bullet$ or $\lambda, \tau(i) \vDash_c \phi_2^\bullet$. By the induction hypothesis of (3), it follows that $\lambda, i \vDash_{pw} \phi_1^\bullet$ or $\lambda, i \vDash_{pw} \phi_2^\bullet$. Therefore, $\lambda, i \vDash_{pw} \phi^\bullet$.

$\phi^\bullet$ **is** $\phi_1^{\star\bullet} \mathcal{R}_{I^\forall} \phi_2^\bullet$**:** By Def. 6.5, $\lambda, \tau(i) \vDash_c \phi^\bullet$ implies that, for all $u \geq \tau(i)$ such that $(u - \tau(i)) \in I^\forall$, either $\lambda, u \vDash_c \phi_2^\bullet$ or there exists r such that $\tau(i) \leq r < u$ and $\lambda, r \vDash_c \phi_1^{\star\bullet}$ (†). Let $k \geq i$ such that $(\tau(k) - \tau(i)) \in I^\forall$. We are in the conditions of applying (†) to $u = \tau(k)$ and, hence, either $\lambda, \tau(k) \vDash_c \phi_2^\bullet$ or there exists r such that $\tau(i) \leq r < \tau(k)$ and $\lambda, r \vDash_c \phi_1^{\star\bullet}$. In the first case, by the induction hypothesis of (3), we have that $\lambda, k \vDash_{pw} \phi_2^\bullet$. In the second case, by the induction hypothesis of (2) we have that there exists j such that $\tau(j) = r$ and $\lambda, j \vDash_{pw} \phi_1^{\star\bullet}$. To conclude that $\lambda, i \vDash_{pw} \phi^{\star\bullet}$, it remains to ensure that $i \leq j < k$, which follows from the fact that $\tau(i) \leq r < \tau(k)$.

$\phi^\bullet$ **is** $\phi_1^\bullet \mathcal{U}_I \phi_2^{\star\bullet}$**:** By Def. 6.5, $\lambda, \tau(i) \vDash_c \phi^\bullet$ implies that there exists $u \geq \tau(i)$ such that $(u - \tau(i)) \in I$, $\lambda, u \vDash_c \phi_2^{\star\bullet}$ and, for all r such that $\tau(i) \leq r < u$, $\lambda, r \vDash_c \phi_1^\bullet$ (†). By the induction hypothesis of (2), there exists k such that $\tau(k) = u$ and $\lambda, k \vDash_{pw} \phi_2^{\star\bullet}$. For this k we have that $(\tau(k) - \tau(i)) \in I$ and $\lambda, k \vDash_{pw} \phi_2^{\star\bullet}$. Let j be such that $i \leq j < k$. Since $\tau(i) \leq \tau(j) < \tau(k)$, we are in the conditions of applying (†) to $r = \tau(j)$ and, hence, $\lambda, \tau(j) \vDash_c \phi_1^\bullet$. By the induction hypothesis of (3), $\lambda, j \vDash_{pw} \phi_1^\bullet$. Hence, we can conclude that $\lambda, i \vDash_{pw} \phi^\bullet$.

□

Proof of Corollary 6.13

If $\lambda \vDash_c \phi^\bullet$ then $\lambda \vDash_{pw} \phi^\bullet$.

Proof. This result is an immediate consequence of (3) of Lemma 6.12. By Def. 6.5, $\lambda \vDash_c \phi^\bullet$ iff $\lambda, 0 \vDash_c \phi^\bullet$. By definition of timed trace $\tau(0) = 0$ and, hence, $\lambda, 0 \vDash_{pw} \phi^\bullet$. By Def. 6.1, we then conclude $\lambda \vDash_{pw} \phi^\bullet$. □

Language of safety-MTL(R)

The language SAFETY-MTL(R) can be defined as follows:

$$\phi^{\circ\bullet} ::= \mathit{true} \mid \mathit{false} \mid a \mid \neg a \mid \phi^{\circ\bullet} \wedge \phi^{\circ\bullet} \mid \phi^{\circ\bullet} \vee \phi^{\circ\bullet} \mid \phi^{\star\circ\bullet}\ \mathcal{R}_{I^<}\ \phi^{-\bullet}$$
$$\mid \phi^{\star\circ\bullet}\ \mathcal{R}_{I^\forall}\ \phi^{\diamond\bullet} \mid \phi^{\star\circ\bullet}\ \mathcal{R}_{I^+}\ \phi^{+\bullet} \mid \phi^{-\bullet}\ \mathcal{U}_I\ \phi^{\star\circ\bullet}$$

with

$$\phi^{\diamond\bullet} ::= \mathit{true} \mid \neg a \mid \phi^{\diamond\bullet} \wedge \phi^{\diamond\bullet} \mid \phi^{\diamond\bullet} \vee \phi^{\circ\bullet} \mid \phi^{\circ\bullet} \vee \phi^{\diamond\bullet}$$
$$\phi^{-\bullet} ::= \mathit{true} \mid \mathit{false} \mid \neg a \mid \phi^{-\bullet} \wedge \phi^{-\bullet} \mid \phi^{-\bullet} \vee \phi^{-\bullet} \mid \phi^{\star\circ\bullet}\ \mathcal{R}\ \phi^{-\bullet} \mid \phi^{-\bullet}\ \mathcal{U}_{I_0^-}\ \phi^{\star\circ\bullet}$$
$$\phi^{+\bullet} ::= \mathit{true} \mid \mathit{false} \mid \neg a \mid \phi^{+\bullet} \wedge \phi^{+\bullet} \mid \phi^{+\bullet} \vee \phi^{+\bullet} \mid \phi^{\star\circ\bullet}\ \mathcal{R}\ \phi^{+\bullet}$$
$$\phi^{\star\circ\bullet} ::= \mathit{false} \mid a \mid \phi^{\star\circ\bullet} \wedge \phi^{\circ\bullet} \mid \phi^{\circ\bullet} \wedge \phi^{\star\circ\bullet} \mid \phi^{\star\circ\bullet} \vee \phi^{\star\circ\bullet}$$

where $I^<$ is of the form $[0,t)$ or $[0,t]$ or $[0,\infty)$, I_0^- is of the form $(0,t)$ or $(0,t]$, I^+ is of the form $[t,\infty)$ or (t,∞), $I^\forall$ is any interval, and I is any bounded interval.

Proof of Proposition 6.16

Let λ be a timed trace over A, and $\delta \in \mathbb{Q}_{>0}$: $\lambda \vDash \mathbb{A}x_\delta$ iff λ refines a δ-timed trace.

Proof. By definition of $\mathbb{A}x_\delta$, $\lambda \vDash \mathbb{A}x_\delta$ iff $\boxed{\delta}(\neg(\mathit{true}\ \mathcal{U}_{(0,\delta)}\ \neg(\wedge_{a\in\mathsf{A}}\neg a)))$. By the semantics of $\boxed{\delta}$, this is equivalent to have that, for all $n \in \mathbb{N}$, there exists j such that $\tau(j) = n\cdot\delta$ and $\lambda, j \vDash \neg(\mathit{true}\ \mathcal{U}_{(0,\delta)}\ \neg(\wedge_{a\in\mathsf{A}}\neg a))$. The last part is equivalent to $\lambda, j \nvDash (\mathit{true}\ \mathcal{U}_{(0,\delta)}\ \neg(\wedge_{a\in\mathsf{A}}\neg a))$. By the semantics of $\mathcal{U}$, this happens iff for all $k \geq j$ such that $0 < \tau(k) - \tau(j) < \delta$, $\lambda, k \vDash \wedge_{a\in\mathsf{A}}\neg a$, or equivalently, for all $a \in \mathsf{A}$, $a \notin \tau(k)$. Hence, we can conclude that $\lambda \vDash \mathbb{A}x_\delta$ iff, for all $n \in \mathbb{N}$, there exists j such that $\tau(j) = n\cdot\delta$ and for all $k \geq j$ such that $0 < \tau(k) - \tau(j) < \delta$, $\tau(k) = \emptyset$.

On the other hand, if $\lambda \preceq_\rho \lambda'$ and $\lambda' = \langle\sigma', \tau'\rangle$ is a δ-timed trace, then for all $n \in \mathbb{N}$, $\tau(\rho(n)) = n\cdot\delta$ and $\sigma'(n) = \sigma(\rho(n))$ and for all k such that $\rho(n) < k < \rho(n+1)$, $\sigma(k) = \emptyset$.

($\Leftarrow$) From what we deduced above, we can easily conclude that if λ refines a δ-timed trace then $\lambda \vDash \mathbb{A}x_\delta$: for each n we take $j = \rho(n)$ since for any $k \geq \rho(n)$, if $0 < \tau(k) - n\cdot\delta < \delta$ then $\rho(n) < k < \rho(n+1)$.

($\Rightarrow$) If $\lambda \vDash \mathbb{A}x_\delta$ then, for all $n \in \mathbb{N}$, there exists j such that $\tau(j) = n \cdot \delta$, which we take as $\rho(n)$. Consider the following δ-timed trace: $\lambda' = \langle \sigma', \tau_\delta \rangle$ with $\sigma'(i) = \sigma(\rho(i))$. Since for all $k \geq \rho(n)$ such that $0 < \tau(k) - n \cdot \delta < \delta$ we have that $\tau(k) = \emptyset$, we can conclude that $\lambda \preceq_\rho \lambda'$. □

Proof of Proposition 6.18

The sentence $\mathbb{A}x_\delta$ defines a divergent safe property.

Proof. By definition, $\mathbb{A}x_\delta = \boxdot(\neg(\mathit{true}\ \mathcal{U}_{(0,\delta)}\ \neg(\wedge_{a\in\mathsf{A}}\neg a)))$. We need to prove that $\Lambda = \{\lambda : \lambda \vDash \mathbb{A}x_\delta\}$ is divergent safe, i.e., that for any timed trace λ, if for all $\pi{<}\lambda$ there is $\lambda'{\in}\Lambda$ such that $\pi{<}\lambda'$, then $\lambda{\in}\Lambda$.

Consider a timed trace λ such that $\lambda{\notin}\Lambda$, i.e., $\lambda \nvDash \mathbb{A}x_\delta$. In this case, there exists $n \in \mathbb{N}$ such that either (1) for all j, $\tau(j) \neq n \cdot \delta$ or (2) exists j such that $\tau(j) = n \cdot \delta$ and exists $k \geq j$ such that $0 < \tau(k) - \tau(j) < \delta$ and $\tau(k) \neq \emptyset$. In case (1), since time progresses in τ, there exists k such that $\tau(k) > n \cdot \delta$.

We consider the prefix $\pi = \langle \sigma_{k+1}, \tau_{k+1} \rangle$ of λ. It is easy to conclude that, in both cases, any timed trace λ' that admits π as prefix also does not satisfy $\mathbb{A}x_\delta$. □

Proof of Theorem 6.21

Let α be a HT-ARN and, for every node (resp. hyperedge) p, let Φ_p be a specification of the process (resp. orchestrator) at p. Let

$$\Phi_\alpha = \bigcup_{p\in N\cup E} \iota_p(\Phi_p \cup \mathbb{A}x_{\delta_p}^{\mathsf{A}_{\gamma_p}})$$

where the functions ι_p translate the sentences in the specification of the process at p (resp. of the orchestrator at c) to the language A_α. We have that $\Lambda_\alpha \vDash \phi$ if $\Phi_\alpha \vdash \phi$.

Proof. Let $\lambda \in \Lambda_\alpha$. We have that $\Lambda_\alpha = \bigcap_{p\in N\cup E} \iota_p(\Lambda_p)$ and, hence, for every $p \in N \cup E$, $\lambda \in \iota_p(\Lambda_p)$. This implies that, for every $p \in N \cup E$, $\lambda_{|\iota_p} \in \Lambda_p$. Since Φ_p is a specification of the process (or orchestrator) at p, $\lambda_{|\iota_p} \vDash \Phi_p$. On the other hand, we also have that $\lambda_{|\iota_p} \vDash \mathbb{A}x_{\delta_p}^{\mathsf{A}_{\gamma_p}}$. Hence we can conclude that, for every $p \in N \cup E$, $\lambda \vDash \iota_p(\Phi_p)$ and $\lambda \vDash \iota_p(\mathbb{A}x_{\delta_p}^{\mathsf{A}_{\gamma_p}})$, i.e., $\lambda \vDash \Phi_\alpha$. By hypothesis $\Phi_\alpha \vdash \phi$ and, hence, $\lambda \vDash \phi$. □